UNIVERSIDAD DE LA RIOJA

TRABAJO FIN DE ESTUDIOS

Título

Diseño, desarrollo y construcción de un robot móvil de tipo péndulo invertido

Autor/es

JORGE BUENO GIL

Director/es

JAVIER RICO AZAGRA y MONTSERRAT GIL MARTÍNEZ ,

Facultad

Escuela Técnica Superior de Ingeniería Industrial

Titulación

Grado en Ingeniería Electrónica Industrial y Automática

Departamento

INGENIERÍA ELÉCTRICA

Curso académico

2018-19

publicaciones.unirioja.es
E-mail: publicaciones@unirioja.es

DISEÑO, DESARROLLO Y CONSTRUCCIÓN DE UN ROBOT MÓVIL DE TIPO PÉNDULO INVERTIDO

TRABAJO FIN DE GRADO

UNIVERSIDAD DE LA RIOJA

GRADO EN INGENIERÍA ELECTRÓNICA INDUSTRIAL Y AUTOMÁTICA

CURSO 2018/19

REALIZADO POR: JORGE BUENO GIL

TUTORES DEL PROYECTO: JAVIER RICO AZAGRA
MONTSERRAT GIL MARTÍNEZ

RESUMEN

Este proyecto consiste en el diseño y control de un sistema electromecánico inestable de dos ruedas. Se basa en el principio del péndulo invertido, un sistema no lineal e inestable que se controla mediante un microcontrolador Arduino, se utiliza también un sensor IMU con giroscopio y acelerómetro para la lectura del ángulo de inclinación del robot, y un driver para la etapa de potencia que controla los motores de corriente continua.

La estructura del robot se realiza con tablerillo de madera y varillas roscadas metalizadas. Por otra parte, para diseñar el controlador PID y para obtener los parámetros de ajuste necesarios se emplea el método heurístico introduciendo dichos parámetros en el software de Arduino donde se realiza el código principal para conseguir la estabilización del robot equilibrista. Posteriormente, con esos parámetros del PID ya calculados se simula su comportamiento en la herramienta Sisotool de Matlab utilizando la función de transferencia del modelo matemático.

ABSTRACT

This project consists in the design and control of an unstable electromechanical system of two wheels. It is based on the inverted pendulum principle, a non-linear and unstable system that is controlled by an Arduino microcontroller, an IMU sensor with gyroscope and accelerometer is also used to read the angle of inclination of the robot, and a driver for the stage of power that controls DC motors.

The structure of the robot is made with wooden board and metallic threaded rods. Furthermore, to design the PID controller and to obtain the necessary adjustment parameters, the heuristic method is used introducing said parameters in the Arduino software where the main code is written to achieve the stabilization of the balancing robot. Subsequently, with those PID parameters already calculated, its behavior is simulated in the Matlab's Sisotool tool using the transfer function of the mathematical model.

AGRADECIMIENTOS

En primer lugar, me gustaría dar las gracias a mis padres porque si no fuera por su apoyo incondicional desde el primer momento este trabajo no hubiera salido adelante. Además, son las personas que me han permitido alcanzar el sueño que tenía desde pequeño que era ser ingeniero.

También me gustaría agradecer todo el apoyo que me han prestado mis amigos y mis compañeros de clase cuando iba progresando con el trabajo y sobre todo cuando surgían problemas.

Por último, no me quiero olvidar de todos los profesores que me han impartido clases durante el grado y que gracias a ellos soy lo que soy hoy en día. Destacando a mis tutores del proyecto que me han ido guiando y ayudando con los problemas que iban surgiendo y que siempre han tenido la puerta del despacho abierta cuando tenía alguna duda.

ÍNDICE

TRABAJO FIN DE GRADO

UNIVERSIDAD DE LA RIOJA

GRADO EN INGENIERÍA ELECTRÓNICA INDUSTRIAL Y AUTOMÁTICA

CURSO 2018/19

REALIZADO POR: JORGE BUENO GIL

TUTORES DEL PROYECTO: JAVIER RICO AZAGRA
MONTSERRAT GIL MARTÍNEZ

ÍNDICE GLOBAL

MEMORIA

TRABAJO FIN DE GRADO

UNIVERSIDAD DE LA RIOJA

GRADO EN INGENIERÍA ELECTRÓNICA INDUSTRIAL Y AUTOMÁTICA

CURSO 2018/19

REALIZADO POR: JORGE BUENO GIL

TUTORES DEL PROYECTO: JAVIER RICO AZAGRA
MONTSERRAT GIL MARTÍNEZ

ÍNDICE MEMORIA

ÍNDICE DE FIGURAS

ÍNDICE DE TABLAS

1 INTRODUCCIÓN

En este Trabajo Fin de Grado se va a realizar el diseño y construcción de un robot equilibrista. El control del robot está basado en el programa Arduino, aunque también se utilizan Matlab y Simulink para el diseño del modelo matemático y del controlador necesario para la estabilización. Lo que se busca con este robot es solucionar el problema del péndulo invertido, un problema muy común en el campo del control, consiguiendo que un sistema inestable se convierta en un sistema estable gracias a la acción generada por los motores tras la lectura del sensor IMU de entrada.

Para que el robot se mantenga en equilibrio en una posición vertical se utiliza un controlador PID digital programado en Arduino y dicho PID funciona como controlador del sistema. La información que es necesaria para poder realizar el control se obtiene del sensor de entrada del sistema. Este sensor es una Unidad de Medida Inercial (IMU) que incorpora un giroscopio y un acelerómetro, y a través de él se calcula el ángulo de inclinación del robot equilibrista.

Lo más costoso y complicado de la realización de este proyecto es la obtención de los parámetros del controlador PID de una manera correcta. Para conseguir estos valores de una manera satisfactoria se utiliza el método heurístico. También se calcula el modelo matemático del sistema y se introduce en la herramienta Sisotool de Matlab junto con el controlador PID. Esta herramienta permite simular el controlador diseñado y ver gráficamente la respuesta del sistema.

Una vez conseguida la estabilidad del robot se podría añadir un módulo de comunicación inalámbrica con el que se puede conectar el robot a un smartphone mediante comunicación por vía bluetooth empleando una pequeña aplicación. De esta manera el usuario tendría la capacidad de controlar el movimiento del robot como si de un juguete teledirigido se tratase.

Otro aspecto importante a considerar es la elaboración de la estructura del robot equilibrista. Inicialmente se diseñó en FreeCAD, un programa de diseño de piezas en 3D, pero ante la dificultad que suponía la impresión de la pieza se optó por realizarla con tablerillo de madera y varillas metálicas roscadas.

Finalmente, este tipo de robot puede ser empleado dentro del mundo de la enseñanza en asignaturas de control por ejemplo como referencia para el diseño de controladores en sistemas inestables y poder observar también cómo afectan los diferentes parámetros de un controlador PID a un sistema en tiempo real. En los colegios o institutos se pueden impartir charlas explicando el funcionamiento de dicho robot puesto que es bastante llamativo que un robot con solo dos ruedas pueda mantener el equilibrio y desplazarse por sí mismo. Esta sería una forma sencilla de introducir a los alumnos más jóvenes en el mundo de la robótica.

2 OBJETO

En este Trabajo Fin de Grado se intenta resolver el problema del péndulo invertido mediante un control digital, como ya se ha comentado anteriormente, en concreto con un algoritmo de control digital robusto implementado en Arduino. Esto conlleva la necesidad de adquirir nuevos conocimientos en el campo del control, en el ámbito de la electrónica digital y en el campo de la programación. Para llevar a cabo este fin, se necesita definir los diferentes objetivos que se pretenden conseguir en los diversos campos. De esta manera, en el campo del control se proponen los objetivos siguientes:

- Entender el funcionamiento y comportamiento de un controlador PID digital y cada uno de los parámetros que lo componen.
- Comprender la implicación de estos parámetros en un sistema real.
- Estudiar, buscar y analizar información para mejorar el sistema de control.
- Estudiar y comprender el entorno de programación de Matlab y Simulink.

En el ámbito de la electrónica, los objetivos que se quieren buscar son los siguientes:

- Escoger los dispositivos y componentes que forman parte del diseño del robot equilibrista.
- Estudiar e investigar las diferentes conexiones entre los dispositivos electrónicos para el correcto funcionamiento del sistema.

Para acabar, a través de la programación se engloban los objetivos anteriormente descritos y se intenta realizar un programa que cumpla con el propósito principal del trabajo. De esta forma, los objetivos serían los siguientes:

- Estudiar y comprender el entorno de programación de Arduino, basado en el lenguaje C.
- Entender los diversos protocolos de comunicación que se utilizan, concretamente I2C.
- Realizar un programa que sea capaz de leer la información del sensor IMU y genere una acción de control para los motores de continua a través del control digital implementado en él.

3 ALCANCE

Este proyecto se centra en conseguir la estabilidad de un robot equilibrista de dos ruedas mediante el uso de un controlador PID.

Primeramente, para conseguir dicho fin hay que realizar una buena estructura que ayude al robot a estabilizarse. Es decir, que el peso esté uniformemente distribuido y que sea una estructura estable donde se puedan colocar de manera sencilla los diferentes dispositivos necesarios para que funcione correctamente el sistema.

También, hay que encontrar un método de control lo suficientemente fiable para que se estabilice el sistema. El controlador óptimo para este tipo de sistemas inestables y no lineales es el PID (Proporcional, Integral y Derivativo). Si se emplea solo un proporcional el sistema oscila mucho y no consigue estabilizarse, con un PI se consigue eliminar el error en régimen permanente, pero es necesaria la componente derivativa que es la que anticipa el comportamiento futuro del error y ajustando los parámetros correctamente se consigue estabilizar el sistema.

Para implementar dicho controlador se va a utilizar el microcontrolador Arduino que es un software libre y bastante intuitivo para programar. Con este programa se va a realizar también el código completo que permita leer los valores del ángulo de inclinación del sensor IMU y mande a los motores de continua la potencia necesaria para conseguir estabilizar el robot equilibrista.

4 PRINCIPIO DE FUNCIONAMIENTO

Un robot equilibrista o auto balanceado, como el que se realiza en este trabajo, se basa en el principio de un péndulo invertido con dos ruedas. Tiene una superficie que debe mantenerse en equilibrio para que el vehículo pueda mantenerse en pie y se pueda mover sobre la superficie.

Este problema del péndulo invertido ha causado mucho interés entre multitud de investigadores a lo largo de muchos años. En la actualidad se está intentando llevar esta idea a una aplicación cada vez más móvil. Algunos ejemplos posibles serían medios de transporte personales como los segways, robots humanoides, sillas de ruedas robóticas…

Para que el robot equilibrista se mantenga en una posición rectilínea vertical, los motores deben contrarrestar la caída del robot. Esto quiere decir que cada vez que el robot se incline hacia un lado u otro, los motores deben ser capaces de actuar de manera que las ruedas giren en el sentido de la inclinación. Para poder realizar esta acción es necesaria una retroalimentación que informe del estado del robot y un factor de corrección que arregle los posibles errores que ocurran. En este caso se emplea una IMU (Unidad de Medida Inercial) MPU-6050 que incluye un giroscopio y un acelerómetro para poder realizar las medidas oportunas. Este sensor es el que informa al microcontrolador Arduino de la orientación del robot.

Figura 1. Modelos de Segway

5 HISTORIA

Los robots auto balanceados son robots capaces de mantener el equilibrio sobre dos ruedas sin que se vean afectados por fuerzas externas. Para conseguir este fin se basan en el principio del péndulo invertido que consiste en un péndulo o varilla que rota libremente sobre uno de sus extremos. Este extremo está unido a una articulación de un carro que puede desplazarse de manera horizontal.

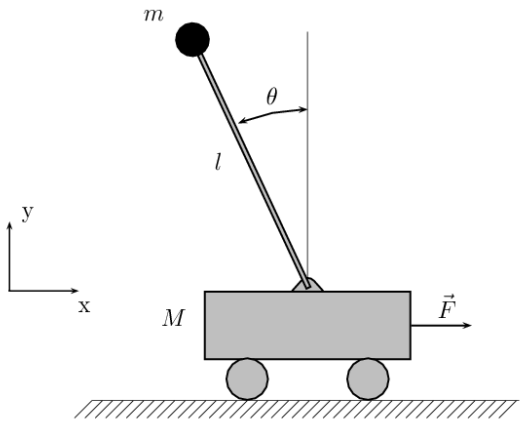


Figura 2. Péndulo invertido

El origen de este tipo de robots está datado en el año 1986 en la Universidad de Electro-Comunicaciones de Chofu, Japón, donde el profesor Kazuo Yamafuji diseñó un robot que era capaz de simular el comportamiento de un péndulo invertido. Dicho robot incluía un eje provisto de dos ruedas y un carro que contenía el dispositivo de estabilización y control.

Figura 3. Robot de Kazuo Yamafuji

Otro ejemplo más reciente de robot que imita al péndulo invertido se puede encontrar en el año 2000. Este robot fue creado en el laboratorio de Electrónica Industrial del Swiss Federal Institute of Technology. Le pusieron pesos al péndulo para simular el peso de un ser humano y así hacerlo más realista. Este prototipo fue llamado Joe y fue una primera versión, la base de los Segway HT.

Figura 4. Joe y Segway HT

A día de hoy los robots equilibristas han aumentado su popularidad considerablemente. Esto es debido principalmente al gran desarrollo que ha habido de plataformas de bajo coste para el desarrollo de sistemas eléctricos y electrónicos, además de la fácil disponibilidad y bajo coste de los componentes electrónicos necesarios para poder ensamblar estos sistemas. De esta manera se pueden encontrar diseños basados en microcontroladores PIC o AVR, e incluso puede llegar a construirse un péndulo invertido mediante Lego y la plataforma Lego Mindstorm, obteniéndose un robot capaz de mantener el equilibrio midiendo la aceleración y calculando su inclinación.

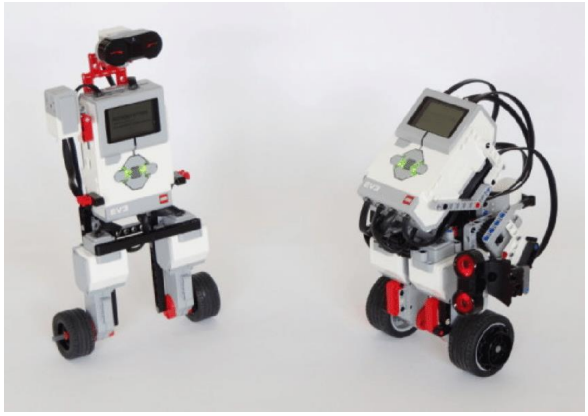


Figura 5. Robots equilibristas de Lego

La principal aplicación comercial de este tipo de robots auto balanceados es el Segway que es un vehículo de transporte eléctrico giroscópico de dos ruedas. Fue creado en 2001 por Dean Kamen y es el primer dispositivo de transporte auto balanceado pensado para el transporte, el trabajo, la seguridad y el ocio. Es normal encontrar tanto patrullas de agentes de seguridad montados sobre Segways como personas normales montadas en ellos y desplazándose por la ciudad, sobre todo en grandes ciudades.

Figura 6. Segway

La idea de futuro de este tipo de robots equilibristas es que sean robots que se mantengan en equilibrio sobre una sola bola. Para ello hay que diseñar un algoritmo de control preciso, así como un sistema de control de tres motores diferentes puesto que el movimiento es omnidireccional. Además, esta forma de desplazamiento necesita un tipo de rueda especial que permita al robot moverse en más de una dirección. Un ejemplo de este tipo de robots podría ser el famoso BB-8 de la película Star Wars.

Figura 7. Robot BB-8

6 APLICACIONES

Este tipo de robots auto balanceados tienen aplicaciones muy variadas. Estas diversas aplicaciones se pueden encontrar en diferentes lugares:

- *Dispositivos comerciales*: Existen empresas que diseñan dispositivos basados en el péndulo invertido para el transporte de personas, mejorando de esta manera la movilidad y la eficiencia a la hora de desplazarse rápidamente al trabajo o cualquier sitio.

Figura 8. Segways

- *Biomecánica*: Suele utilizarse para modelar bípedos caminantes como por ejemplo el caminante dinámico Cassie. En la Oregon State University han desarrollado este robot de curioso e inquietante aspecto que parece un avestruz sin cuerpo o algún engendro mecánico salido de la guarida del Doctor Infierno. En los robots bípedos la pierna de apoyo en contacto con el suelo se modela como un péndulo invertido mientras que la pierna en movimiento se comporta como un péndulo que oscila libremente suspendido de la cadera del humanoide.

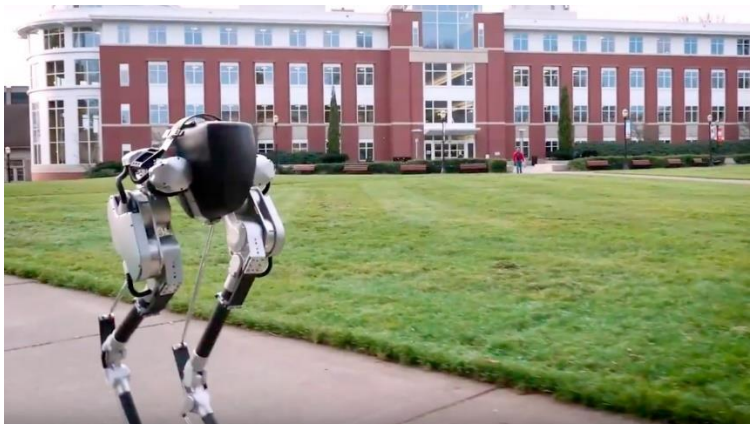


Figura 9. Cassie

- *Laboratorios y aulas*: Este tipo de dispositivos son un banco de pruebas muy útil y completo para la ingeniería de control. Pueden encontrarse diferentes prototipos de péndulo invertido ya sean del tipo péndulo invertido sobre carro o péndulo de Furuta.

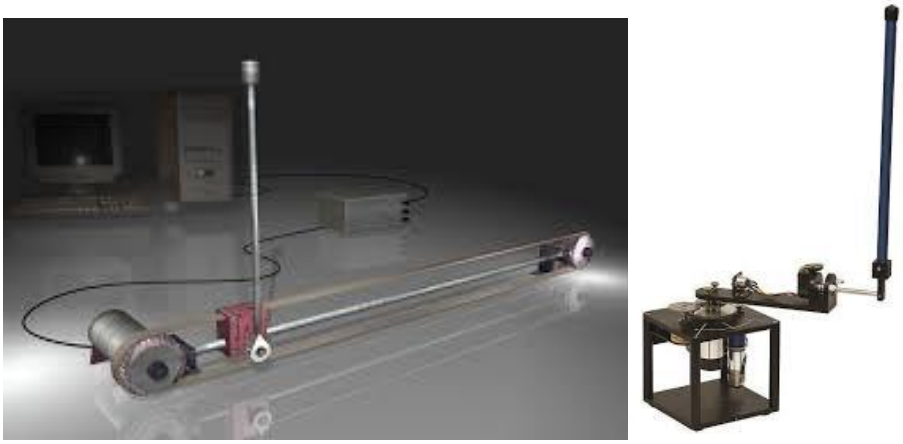


Figura 10. Péndulo invertido sobre carrro y Péndulo de Furuta

- *Aeroespacial*: Es necesario el control activo de un cohete para mantenerlo en la posición vertical invertida durante el despegue. El ángulo de inclinación del cohete es controlado a través de la variación del ángulo de aplicación de la fuerza de empuje. Uno de los ejemplos más conocidos en este sector aeroespacial es el Falcon 9 de la empresa SpaceX. El objetivo es conseguir un cohete de bajo coste, de forma que el cohete sea reutilizable. Por tanto, se busca recuperar la lanzadera intacta sin que reciba daños. Esto se consigue haciendo que la lanzadera aterrice verticalmente por lo que hay que buscar solucionar el problema del péndulo invertido.

Figura 11. Despegue de un cohete Falcon 9

7 DESCRIPCIÓN DEL SISTEMA

El hardware es la parte física del sistema, es decir, todos los componentes eléctricos, electrónicos, electromecánicos y mecánicos que componen el robot equilibrista. A continuación, se van a describir los diversos componentes necesarios para la realización de este proyecto.

7.1 SISTEMA MECÁNICO

El diseño general del robot auto balanceado es un cuerpo con forma de prisma rectangular sobre dos ruedas paralelas entre sí. El prisma rectangular se compone de cuatro bases de tablerillo de madera separadas uniformemente entre ellas como si fuera una estantería, unidas a través de cuatro varillas roscadas situadas en las cuatro esquinas de cada una de las bases de madera.

Inicialmente esta estructura del robot se realizó en 3D en un programa de diseño para ordenador llamado FreeCAD. El diseño había quedado bastante óptimo, pero a la hora de ir a imprimirlo se vio que iba a resultar complicada la impresión en 3D por la forma de la pieza y los orificios que contenía. Por tanto, se optó por la realización de la estructura en tablerillo de madera.

En la base inferior de madera, que se encuentra a la altura del eje de las ruedas y entre los dos motores de continua, están colocadas las seis pilas de 1,5V cada una.

En la siguiente base se encuentra el driver L298N que alimenta a los dos motores. En la tercera base está situado el microcontrolador Arduino y en la última base de madera que sería el techo de la estructura se encuentra la IMU MPU-6050. Se ha colocado el sensor en esa posición puesto que las demás bases disponen de poco espacio libre y así puede realizar las mediciones de manera precisa y correcta.

En la siguiente imagen se puede ver la estructura del robot anteriormente descrita.

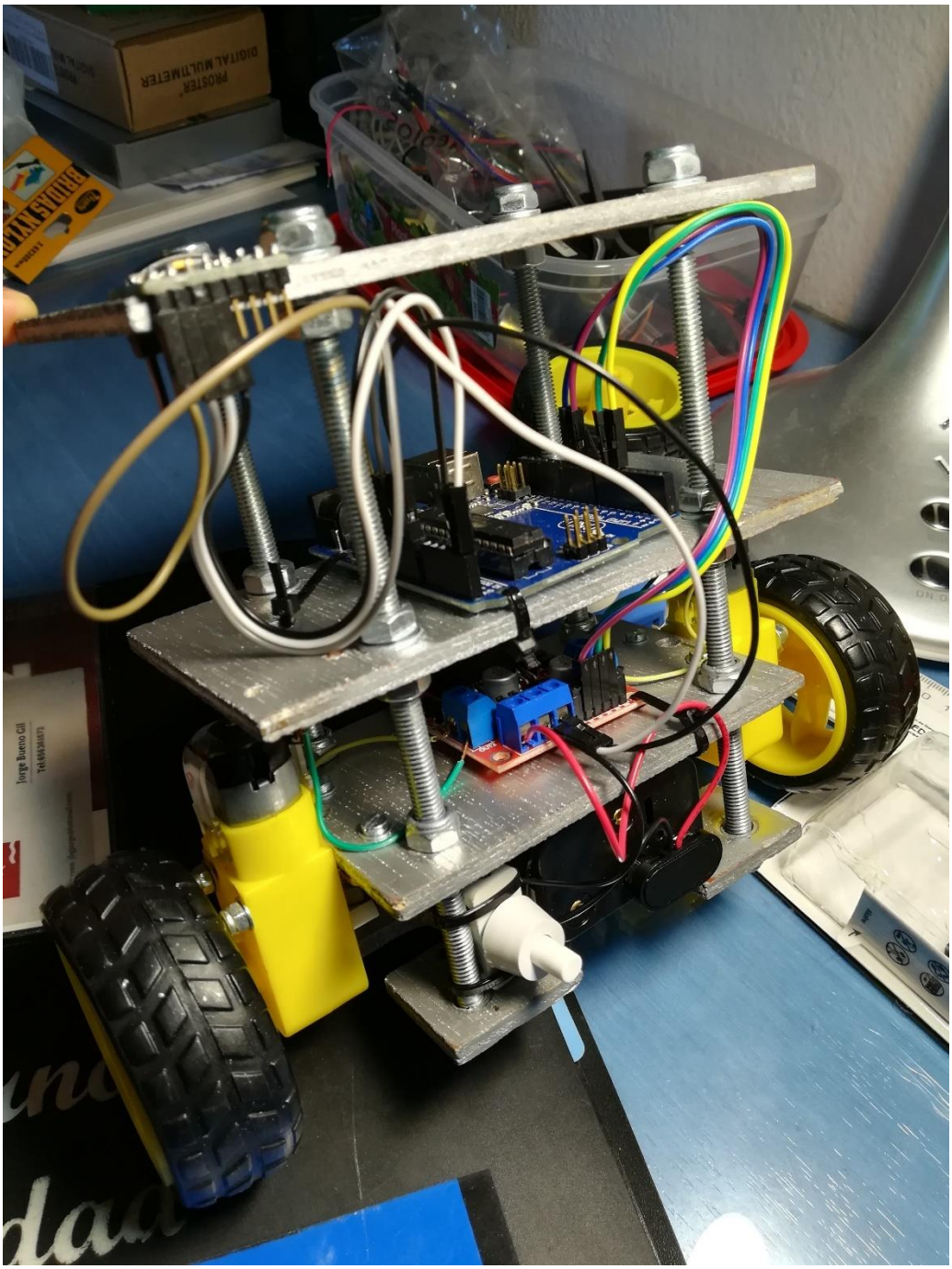


Figura 12. Estructura del robot equilibrista

La anchura de la estructura es de 15cm, la altura de 16cm y la profundidad de 8cm.

Después de haber explicado el montaje mecánico con una descripción de la estructura del robot, sus medidas y la colocación de los componentes, se va a explicar de manera más detallada cada uno de los componentes mecánicos que la forman.

VARILLAS ROSCADAS

Las varillas roscadas metálicas de 6mm de diámetro son las encargadas de unir las cuatro bases de tablerillo de madera, formando así la estructura del robot auto balanceado en forma de prisma rectangular. La longitud de cada una de las varillas es de 16cm y se sitúan en las respectivas esquinas de cada superficie rectangular de madera. Cada base se ajusta con ocho tuercas y ocho arandelas, cuatro en la parte superior y cuatro en la parte inferior respectivamente. En las bases de madera que hacen de techo y de suelo, se han puesto en la parte exterior de ellas tuercas de seguridad por motivos de una mejor sujeción y estética. La segunda superficie rectangular está sujetada también con dos pletinas y cuatro tornillos para dar más seguridad y estabilidad a la estructura del robot.

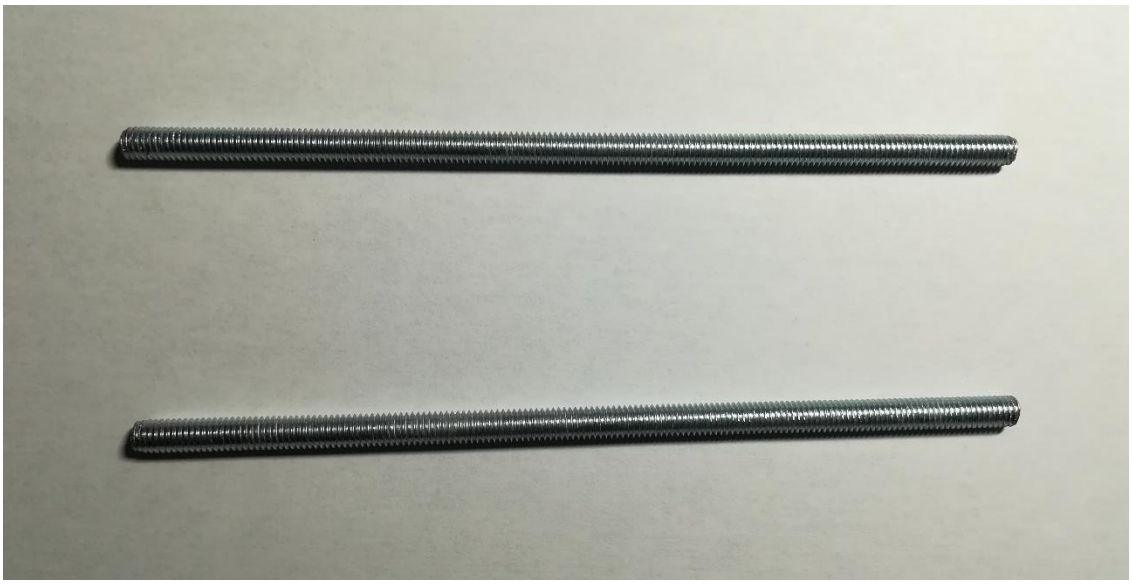


Figura 13. Varillas roscadas

TABLERILLO DE MADERA

Se ha utilizado un tablerillo de madera de 5mm de grosor para las cuatro bases de la estructura del robot. Las dos bases superiores tienen 15cm de anchura y 8cm de profundidad. La segunda base tiene 11cm de anchura y 8cm de profundidad. Y la base inferior tiene 10cm de anchura y 8cm de profundidad. Las cuatro superficies han sido agujereadas para poder pasar las varillas roscadas, el cableado necesario y también poder sujetar de manera sencilla los diferentes componentes.

Figura 14. Tablerillo de madera

RUEDAS

Las ruedas del robot tienen un diámetro de 6cm y una anchura de 2,5cm. Los neumáticos proporcionan al robot una mejor estabilidad teniendo en cuenta las dimensiones de la estructura del robot y la velocidad angular que generan los motores de continua. Dichos neumáticos están compuestos de un plástico ligero que permite un buen agarre y se unen al eje del motor por la parte interior central.

Figura 15. Ruedas

7.2 SISTEMA ELECTRÓNICO

En esta parte del trabajo se va a explicar detalladamente cada uno de los componentes eléctricos que hay instalados en el robot, es decir, los que forman parte de la zona de control, los que están destinados a la adquisición de señales y la etapa de potencia que alimenta a los motores de continua los cuales mueven las ruedas del robot.

7.2.1 UNIDAD DE MEDIDA INERCIAL (IMU)

Las unidades de medida inercial (IMU) son dispositivos electrónicos que miden e informan sobre la velocidad, aceleración y orientación de un dispositivo. Para ello emplean la información combinada de diversos sensores, normalmente acelerómetros, giroscopios y magnetómetros. Este tipo de dispositivos son el principal componente de los sistemas de navegación usados en aviones, buques, naves espaciales, misiles guiados…

La mayor parte de las IMUs se basan en la combinación de un acelerómetro en tres ejes y un giroscopio en tres ejes. Estos dos dispositivos se complementan muy bien puesto que entre ambos se compensan las limitaciones de cada uno. Dichas limitaciones son las siguientes:

- Los giroscopios funcionan bien cuando hay movimientos cortos o bruscos. El problema que tienen es que miden la velocidad angular por lo que necesitan obtener el ángulo por integración respecto al tiempo, surgiendo así errores y ruido en la medición. Este es el motivo por el que aparece el error a medio o largo plazo.
- Los acelerómetros no tienen error a medio o largo plazo puesto que miden el ángulo absoluto que forma el sensor con la dirección vertical marcada por la gravedad. Sin embargo, son muy sensibles a los movimientos del sensor, por tanto, puede aparecer una gran cantidad de ruido de alta frecuencia haciendo que las medidas no sean fiables a corto plazo.

Se puede observar que las características de medición entre los giroscopios y los acelerómetros son opuestas, pero se complementan muy bien entre ellos y de esta manera se obtienen medidas más fiables que usándolos por separado. Así que para usar una IMU de manera correcta es necesario combinar

las mediciones de ambos dispositivos. Para combinar ambos sensores suelen emplearse dos métodos diferentes:

- ✓ Filtro de Kalman: Es un algoritmo que fue desarrollado en 1960 por Rudolf E. Kalman y permite la combinación de información de diferentes fuentes ruidosas minimizando el error mediante un bucle predictivo. A pesar de que se obtienen mejores medidas su coste computacional es mucho más elevado.
- ✓ Filtro complementario: Se basa en combinar un filtro paso alto y otro paso bajo. Es fácil su implementación y tiene un bajo coste computacional.

El objetivo final de todo esto es obtener una medida lo más fiable y realista posible de los ángulos de navegación del objeto. Los ángulos de navegación permiten expresar la orientación relativa de ambos sistemas, describiendo la orientación de un objeto mediante tres rotaciones ortogonales en torno al eje X, Y y Z.

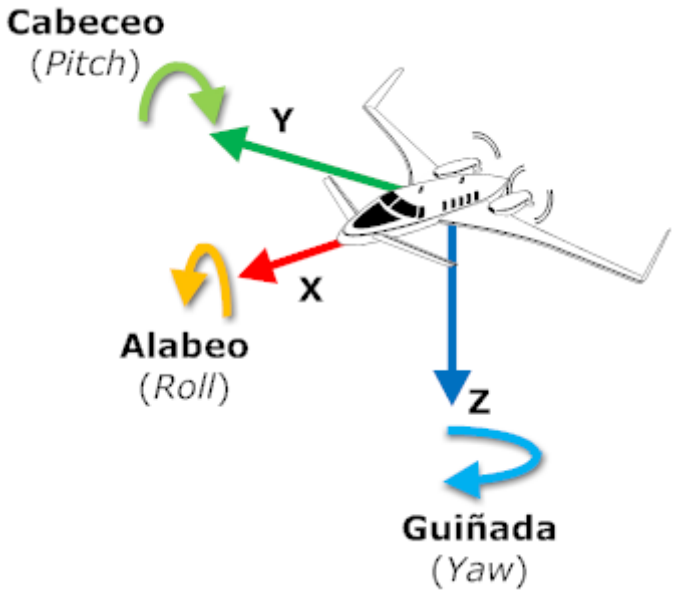


Figura 16. Ángulos de navegación

7.2.1.1 GIROSCOPIO

El giroscopio es un dispositivo que sirve para medir el ángulo de rotación girado por un determinado mecanismo. Este tipo de dispositivos son puramente diferenciales, es decir, siempre miden ángulos relativos respecto a una referencia arbitraria.

El giroscopio que se va a emplear en este proyecto es un giroscopio de estructura vibrante. Es un tipo de giroscopio que funciona de manera similar a los halterios presentes en algunos insectos. El principio físico en el que se basa es que un objeto vibrante tiende a continuar vibrando en el mismo plano que gira su apoyo. En ingeniería este tipo de dispositivos se conocen como giroscopios de vibración de Coriolis. El efecto Coriolis es una fuerza que aparece sobre un cuerpo en movimiento cuando este se encuentra en un sistema de rotación. Debido a este efecto el objeto vibratorio ejerce una fuerza y midiéndola se puede determinar la rotación a la que está sometido el giroscopio.

Para que se consiga registrar el efecto de la fuerza Coriolis, unas partes del giroscopio se someten a vibraciones debido a la rotación del giroscopio y el efecto que ejerce la fuerza de Coriolis deforma la estructura, lo cual provoca que la capacitancia del sistema varíe.

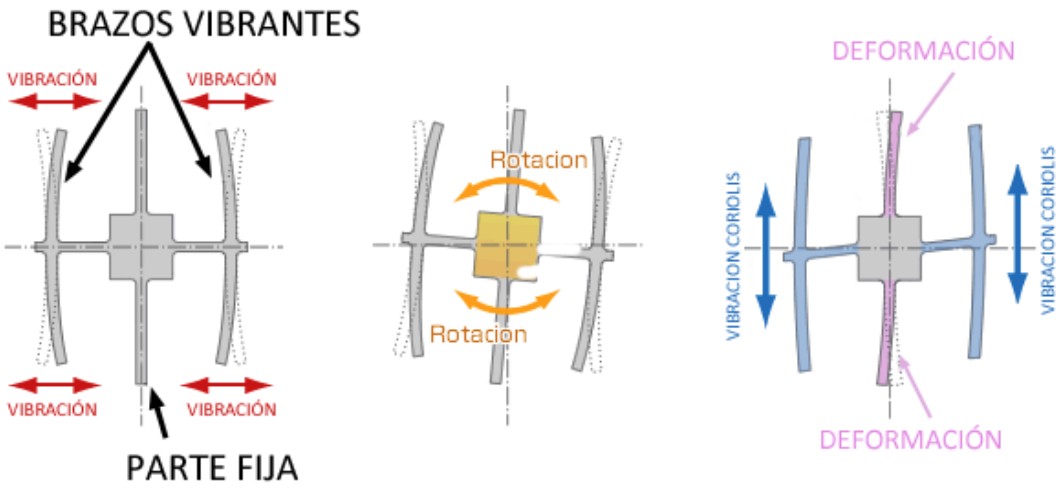


Figura 17. Funcionamiento del giroscopio

El giroscopio proporciona información de tres ejes, es decir, se obtiene la rotación en X, Y y Z de forma independiente (en los tres ángulos de navegación). Sin embargo, como se usa la fuerza de Coriolis para realizar la medida, los

giroscopios vibratorios no registran el ángulo girado sino la velocidad angular (variación del ángulo respecto al tiempo). Para obtener el ángulo de posición del sensor se necesita realizar la integración respecto al tiempo.

El problema principal en estos giroscopios proviene de esta integración ya que se va a producir una acumulación de errores en la medida lo que va a causar una deriva en la medición a medio y largo plazo respecto a su valor real. Sin embargo, la ventaja de estos dispositivos es que son sensores con una respuesta rápida y gran precisión en tiempos cortos, además de presentar una inmunidad al ruido decente. Pero todo esto es válido únicamente para pequeños intervalos de tiempo.

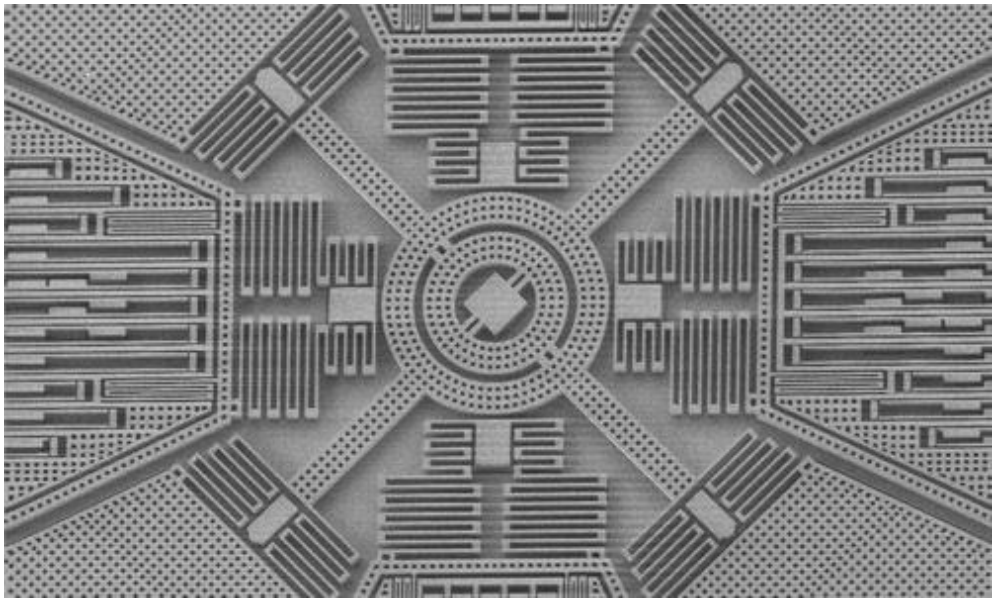


Figura 183. Giroscopio vibratorio

7.2.1.2 ACELERÓMETRO

Este dispositivo está destinado a la obtención de medidas absolutas de las aceleraciones en los tres ejes, X, Y y Z.

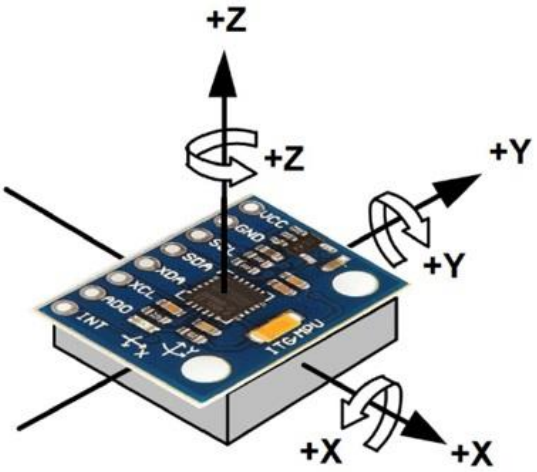


Figura 19. Ejes del sensor MPU 6050

Para la construcción de los dispositivos capaces de medir la aceleración se emplea la primera ley de Newton. Esta ley dice que cualquier cuerpo que tenga una masa cualquiera, requiere una fuerza para poder variar su velocidad. De manera análoga, cualquier cuerpo que esté sometido a una aceleración experimentará cierta fuerza.

Para fabricar un sensor de aceleración se utiliza un cuerpo sólido y en su interior se suspende una masa sujeta por muelles al cuerpo exterior. Aplicando una aceleración a este conjunto, la masa suspendida se desplaza ejerciendo una fuerza sobre los muelles. Lo que provoca esta fuerza es que uno de esos muelles se estire y otro se contraiga.

Para la construcción del acelerómetro en este proyecto se emplea una estructura de polisilicio micromecanizada construida sobre una oblea de silicio. Los muelles de polisilicio suspenden la estructura sobre la oblea y le proporcionan resistencia para soportar las aceleraciones.

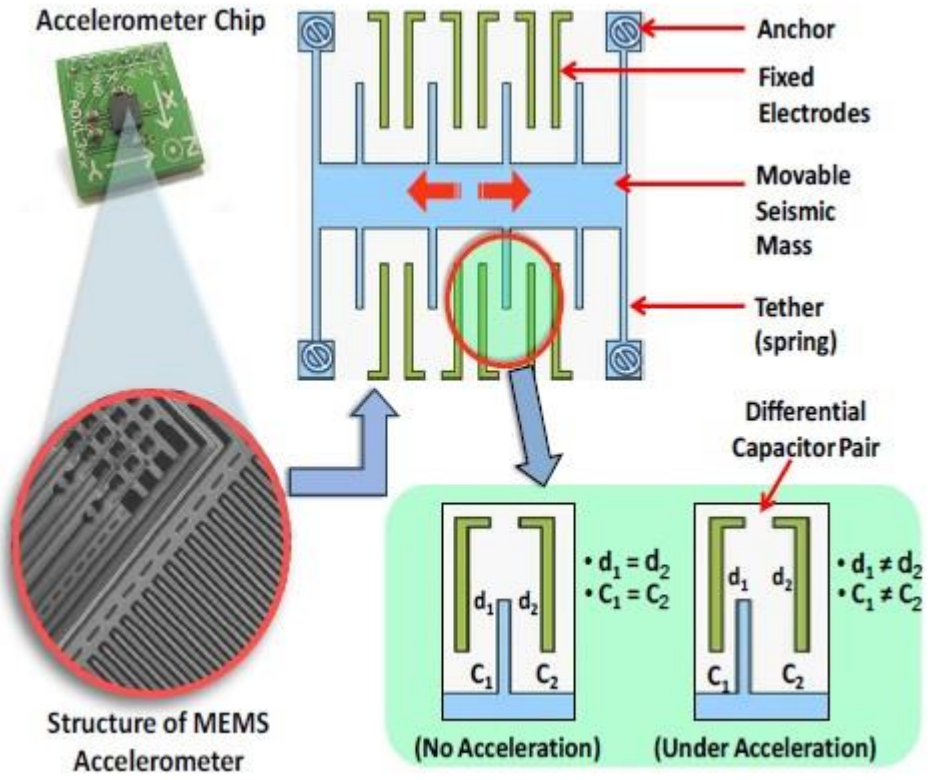


Figura 20. Estructura del acelerómetro

Al someter al dispositivo a aceleraciones la parte inferior se deforma y se desplaza respecto a la parte fija. Este desplazamiento se detecta mediante la variación de la capacidad del sistema. La mayoría de los acelerómetros son de tres ejes por lo que son capaces de medir la aceleración en X, Y y Z independientemente. Así se puede conocer la magnitud y dirección de la aceleración de forma absoluta simultáneamente.

Para comprender esto mejor suponemos que el acelerómetro es un cubo con una bola en su interior. Cada cara del cubo es una dirección del espacio y la cara opuesta la dirección negativa. Si nos colocamos en un lugar que no esté afectado por la gravedad, la aceleración a la que está sometida la bola del interior es cero y por tanto la aceleración en los tres ejes también es cero. Si movemos el cubo hacia la izquierda con una aceleración de 1g la bola se apoyará sobre la cara X negativa y el valor de la aceleración en el eje X será -1g. Por otra parte, si colocamos el acelerómetro inclinado 45 grados a la derecha, la bola tocará dos caras a la vez.

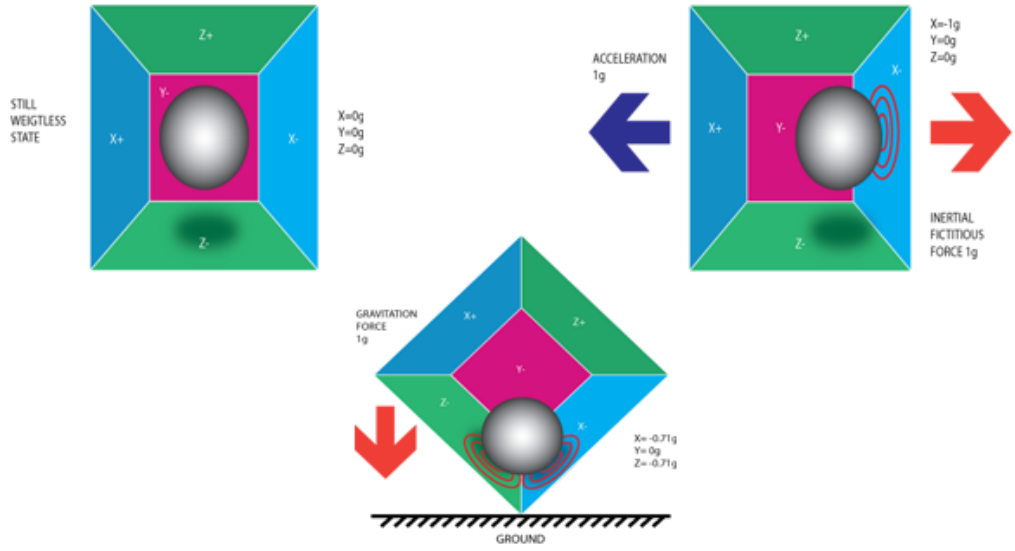


Figura 21. Funcionamiento del acelerómetro

Para el cálculo del ángulo consideramos el eje de coordenadas fijado a los ejes del acelerómetro y el vector fuerza rota sobre el origen del centro de coordenadas. Con las siguientes fórmulas calculamos el ángulo del sensor:

$$\theta_x = \tan^{-1} \frac{A_x}{\sqrt{{A_y}^2 + {A_z}^2}}$$

$$\theta_y = \tan^{-1} \frac{A_y}{\sqrt{{A_x}^2 + {A_z}^2}}$$

Por otra parte, el acelerómetro no es un sensor adecuado para determinar la velocidad de un sistema ni tampoco su posición.

7.2.1.3 FILTRO COMPLEMENTARIO

El giroscopio permite realizar medidas precisas y rápidas, pero presenta una deriva en la medida. Por el contrario, el acelerómetro presenta una respuesta sin deriva, pero se ve muy influenciado por el ruido. Para la compensación de ambos errores se puede emplear la medida obtenida por el giroscopio para tiempos cortos y corregir la deriva con la medida realizada por el acelerómetro en tiempos largos. Esto se puede conseguir mediante un filtro complementario.

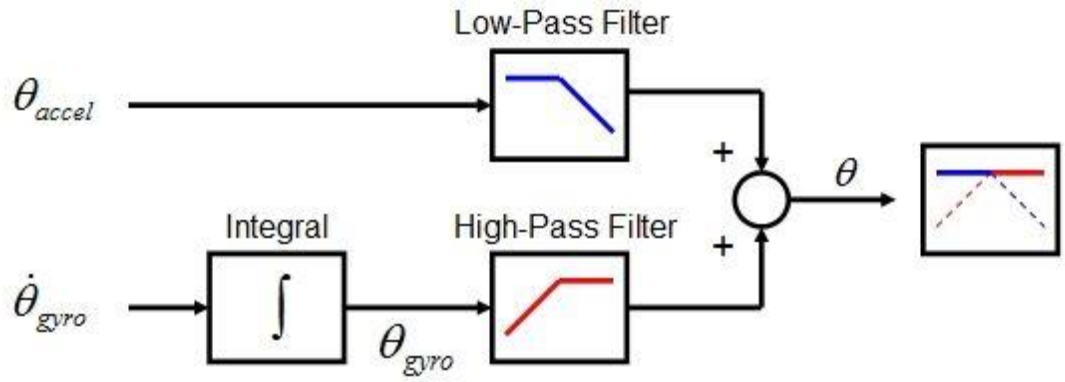


Figura 22. Esquema del filtro complementario

El filtro complementario es un filtro de Kalman de estado estacionario para un cierto tipo de problemas de filtrado. El concepto básico de este filtro es combinar las dos salidas obtenidas con el giroscopio y el acelerómetro y así conseguir una buena aproximación del ángulo de orientación. El filtro funciona como un filtro paso-bajo para medir el acelerómetro y como un filtro paso-alto para medir el giroscopio. De esta forma las componentes de alta frecuencia del acelerómetro están dominadas por el giroscopio. La fórmula matemática para implementar este filtro es:

$$ang_{final} = K * ang_{giro} + (1 - K) * ang_{acel}$$

K es una constante definida por el usuario, ang_{giro} es el ángulo obtenido con el giroscopio, ang_{acel} es el ángulo obtenido con el acelerómetro y ang_{final} es el ángulo final combinado entre giroscopio y acelerómetro.

En la siguiente imagen se puede ver un ejemplo de aplicación del filtro complementario.

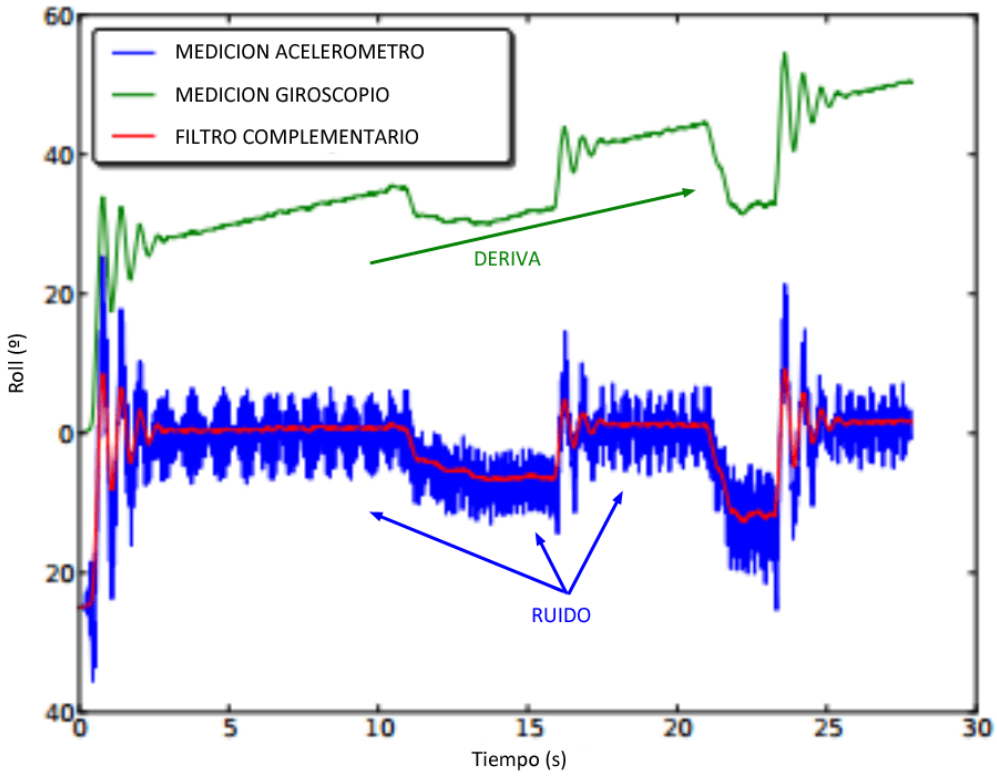


Figura 23. Deriva producida por el giroscopio y ruido del acelerómetro

Se puede observar que el giroscopio presenta una deriva lenta en el tiempo que hace que las medidas sean válidas solo a corto plazo. El acelerómetro presenta una gran cantidad de ruido, aunque las medidas permanecen estables tiempos largos. Usando el filtro complementario se consigue eliminar estos efectos obteniendo una señal sin ruido y estable en el tiempo.

7.2.1.4 COMUNICACIÓN POR I2C CON ARDUINO

I2C es un protocolo de comunicación síncrono y solo emplea dos cables, uno para el reloj (SCL) y otro para el dato (SDA). Esto quiere decir que el maestro y el esclavo envían datos por el mismo cable, el cual es controlado por el maestro que crea la señal de reloj. Este protocolo de comunicación no utiliza selección de esclavo sino direccionamiento.

I2C es también un bus de comunicación en serie. Su nombre viene de Inter-Integrated Circuit y su diseñador es Philips. La velocidad de trabajo es de 100 kbit/s en el modo estándar, aunque también admite velocidades de 3.4 Mbit/s. Es un tipo de bus muy utilizado en la industria, sobre todo para comunicar microcontroladores y sus periféricos en sistemas integrados.

La característica principal del bus I2C es que emplea dos líneas para transmitir la información como ya se ha comentado anteriormente, una para los datos y otra para la señal de reloj. También se necesita una tercera línea, pero esta solo es la masa. Las dos primeras líneas son drenador abierto, por lo tanto, necesitan resistencias de pull-up. A través del mismo cable, dos o más señales pueden causar conflicto, por ejemplo, si se envía un 1 y un 0 lógico a la vez. Para que no ocurra esto el bus I2C es cableado con dos resistencias para poner el bus a nivel alto y los dispositivos conectados envían niveles bajos. Si se quiere enviar un nivel alto hay que comunicarlo al bus.

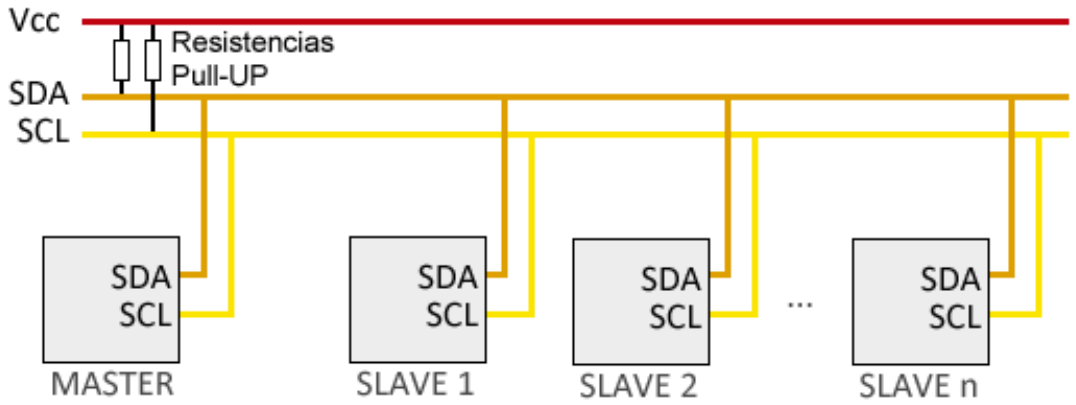


Figura 24. Comunicación por bus I2C

Los dispositivos conectados al bus tienen una dirección única para cada uno y pueden ser maestros o esclavos. En nuestro caso el maestro sería el microcontrolador Arduino y el esclavo el sensor IMU. El maestro inicia la

transferencia de datos y genera la señal de reloj, pero no tiene por qué ser siempre el mismo dispositivo. Esto hace que al bus I2C se le llame bus multimaestro.

El proceso de comunicación en el bus I2C es el siguiente:

- El maestro empieza la comunicación enviando un comando llamado start condition. Esto alerta a los esclavos que se ponen a la espera.
- El maestro se dirige al dispositivo con el que se quiere comunicar, enviando un byte que contiene los siete bits que componen la dirección del esclavo con el que quiere hablar y el octavo bit de menor peso se corresponde con la operación deseada (lectura = 1 o escritura = 0).
- La dirección enviada es comparada por cada esclavo con su propia dirección para ver si coinciden. Cuando coincidan ese será el esclavo direccionado.
- Cada byte leído/escrito por el maestro debe ser reconocido por un bit de ACK por el dispositivo maestro/esclavo.
- Cuando se acaba la comunicación, el maestro envía una stop condition para dejar libre el bus I2C.

Escritura de un dato:

Figura 25. Escritura de un dato por I2C

Lectura de un dato:

Figura 26. Lectura de un dato por I2C

Cuando los datos son enviados por SDA, los pulsos de reloj son enviados por SCL para mantener el maestro y el esclavo sincronizados. Los datos se envían como un bit en cada pulso de reloj.

Arduino dispone de soporte I2C por hardware vinculado físicamente a ciertos pines. También se puede emplear cualquier otro grupo de pines como bus I2C mediante software, pero en ese caso la velocidad sería mucho menor. Los pines asociados en Arduino Uno son los siguientes:

- SDA --> A4
- SCL --> A5

Para utilizar el bus I2C en Arduino, el IDE de Arduino proporciona la librería Wire.h que contiene las funciones necesarias para controlar el hardware integrado.

En el apartado Programa de Arduino se puede ver el código que es necesario implementar para poder leer correctamente los datos del sensor IMU.

7.2.1.5 ELECCIÓN DE LA IMU MPU-6050

Para la realización de este proyecto se ha estado dudando entre dos modelos de IMU, la IMU MPU-6050 y la IMU MPU-9150.

Esta segunda está formada por la primera y además un magnetómetro AK8975. Es una IMU por tanto de 9DOF en vez de 6DOF como es la MPU-6050. La comunicación se puede realizar tanto por bus SPI como por bus I2C mientras que en la MPU-6050 solo se puede realizar por I2C. También la MPU-9150, al incorporar un magnetómetro elimina la deriva que puede aparecer en otras IMUs al cabo de unas horas.

A pesar de todo esto con el giroscopio y el acelerómetro de la MPU-6050 nos sirve para nuestro proyecto y la comunicación por I2C es más sencilla de realizar que por SPI. Además, en cuestión económica sale más rentable de precio la MPU-6050 que la MPU-9150.

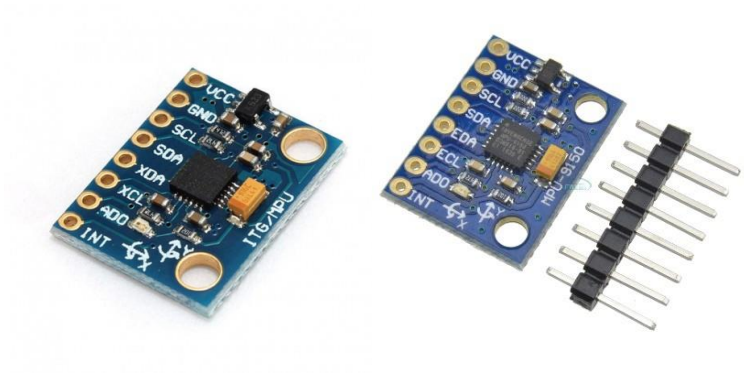


Figura 27. A la izda. IMU MPU-6050 y a la dcha. IMU MPU-9150

7.2.2 MOTORES DE CORRIENTE CONTINUA

7.2.2.1 INTRODUCCIÓN

El motor de corriente continua es una máquina que convierte energía eléctrica en mecánica provocando un movimiento rotatorio gracias a la acción de un campo magnético.

Los motores de continua se componen esencialmente de dos partes. El estátor o inductor se encarga de dar soporte mecánico al aparato y contiene los polos de la máquina que pueden ser devanados de hilo de cobre sobre un núcleo de hierro o imanes permanentes. El rotor o inducido suele ser de forma cilíndrica también devanado y con núcleo. Es alimentado mediante corriente continua a través de delgas las cuales están en contacto alternante con escobillas fijas. Es la parte más frágil del motor.

El problema principal de estas máquinas es el mantenimiento, muy costoso y laborioso, debido al desgaste que sufren las escobillas al estar en contacto constantemente con las delgas.

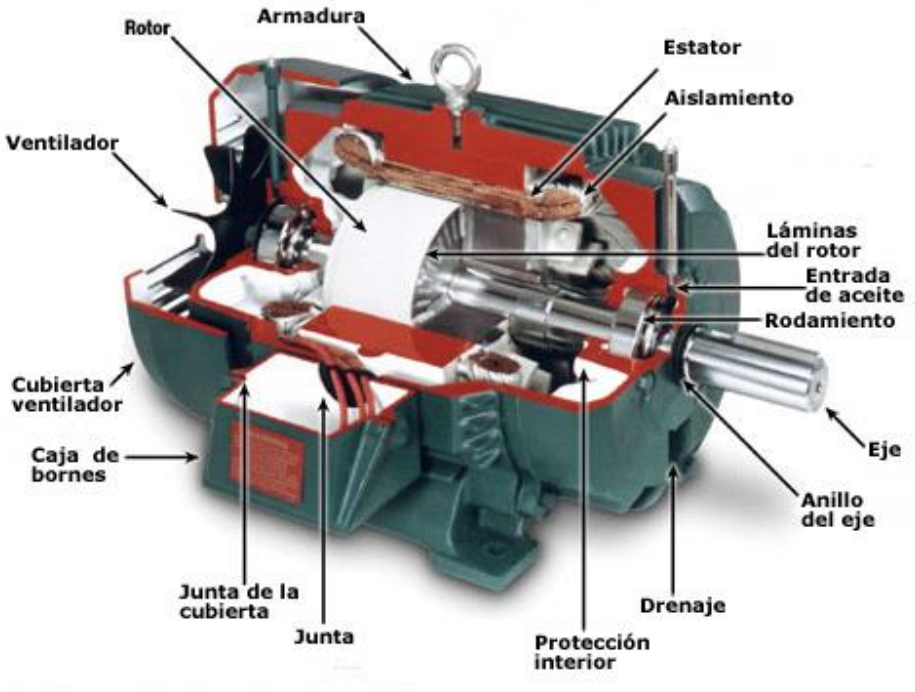


Figura 28. Partes de un motor de continua

7.2.2.2 PRINCIPIO DE FUNCIONAMIENTO

Cuando un conductor por el cual pasa una corriente eléctrica se introduce en un campo magnético, dicho conductor experimenta una fuerza perpendicular al plano formado por el campo magnético y la corriente, de acuerdo con la Fuerza de Lorentz:

$$F = B \cdot L \cdot I \cdot sen\ \emptyset$$

F: Fuerza en newtons.

B: Densidad de campo magnético en teslas.

L: Longitud del conductor en metros.

I: Intensidad que recorre el conductor en amperios.

Φ: Ángulo que forma I con B.

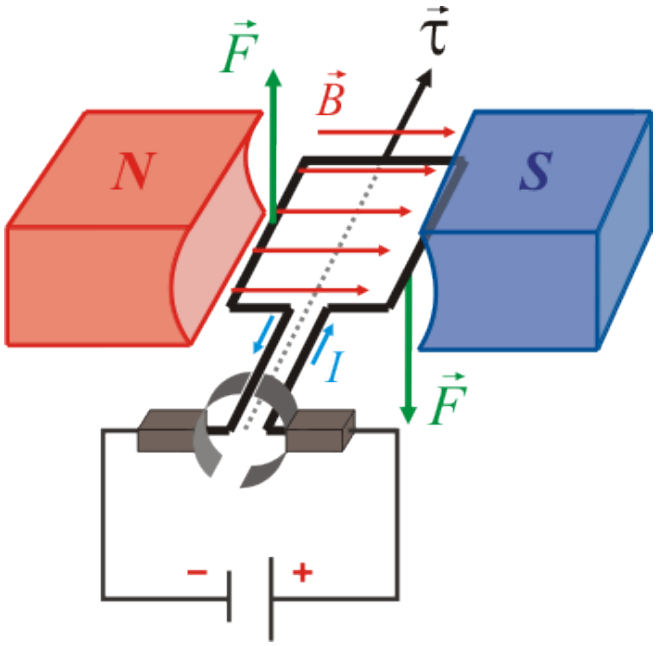


Figura 29. Explicación de la Fuerza de Lorentz

Un motor de corriente continua produce un par debido a la conmutación mecánica de la corriente. En la imagen anterior existe un campo magnético permanente producido por imanes en el estátor. El flujo de corriente en el devanado del rotor produce una fuerza de Lorentz sobre el devanado representada en verde. Como el motor tiene dos polos, la conmutación se realiza a través de un anillo partido a la mitad donde el flujo de corriente se invierte cada media vuelta.

La fuerza contraelectromotriz inducida en un motor es la tensión que se genera en los conductores de un motor debido al corte de las líneas de fuerza. La polaridad de la tensión en los generadores es opuesta a la aplicada en los bornes del motor. En el arranque de un motor de corriente continua se producen fuertes picos de corriente puesto que, al estar la máquina parada, no existe fuerza contraelectromotriz y el bobinado se comporta como un simple conductor de baja resistencia. La fuerza contraelectromotriz en un motor depende directamente de la velocidad de giro del motor y del flujo magnético del sistema inductor.

Las escobillas deben poner en cortocircuito todas las bobinas colocadas en la zona neutra. Si la máquina tiene dos polos, va a haber también dos zonas neutras. Por tanto, el número total de delgas ha de ser igual al número de polos de la máquina. En lo que se refiere a su posición, será coincidente con las líneas neutras de los polos.

7.2.2.3 ELECCIÓN DE LOS MOTORES DE CORRIENTE CONTINUA

En lo que se refiere a motores, para este proyecto se ha pensado en tres tipos diferentes: motores paso a paso, servomotores o motores DC.

Los servomotores son un conjunto de cuatro cosas: un motor de corriente continua, un conjunto de engranajes, un circuito de control y un sensor de posición que puede ser un potenciómetro. La posición de los servos se puede controlar con mayor precisión que la de un motor de continua estándar. Los servos no giran libremente, sino que el ángulo de rotación se limita a 180 grados de ida y vuelta. Además, este tipo de motores tienen un alto par rápido y rotación precisa dentro de un ángulo limitado.

Los motores paso a paso son servos con un método diferente de motorización. Los servos utilizan la rotación continua de un motor DC controlando el giro con el circuito de control integrado mientras que los motores paso a paso emplean múltiples electroimanes dispuestos alrededor de un engranaje central para definir su posición. Los motores paso a paso no se conectan directamente a Arduino lo cual es un problema, requieren un circuito de control externo. Además, estos motores pueden girar 360 grados y son más lentos que los servos.

Para este proyecto no se requiere mucho par ni tanta precisión en las medidas por tanto con motores de corriente continua estándar sirve perfectamente. Estos motores aportan la potencia necesaria para mover el robot y además económicamente son más baratos que los servos o los paso a paso.

Figura 30. De izda. a dcha. motor DC, motor paso a paso y servomotor

7.2.3 DRIVER L298N

El módulo controlador de motores L298N permite controlar la velocidad y la dirección de dos motores de corriente continua de una manera muy fácil, gracias a los dos puentes en H que contiene.

Un puente en H es un componente formado por cuatro transistores que permite invertir el sentido de la corriente y de esta forma se puede invertir el sentido de giro del motor.

El rango de tensiones en el cual funciona este módulo va desde 3V hasta 35V y una corriente de hasta 2A. Cuando se vaya a alimentar el módulo hay que tener en cuenta que su propia electrónica consume unos 3V, por tanto, los motores van a recibir 3V menos que la tensión de partida.

El L298N incluye también un regulador de tensión que permite obtener del módulo una tensión de 5V, idónea para poder alimentar el Arduino. Dicho regulador solo funciona si se alimenta el módulo con una tensión máxima de 12V.

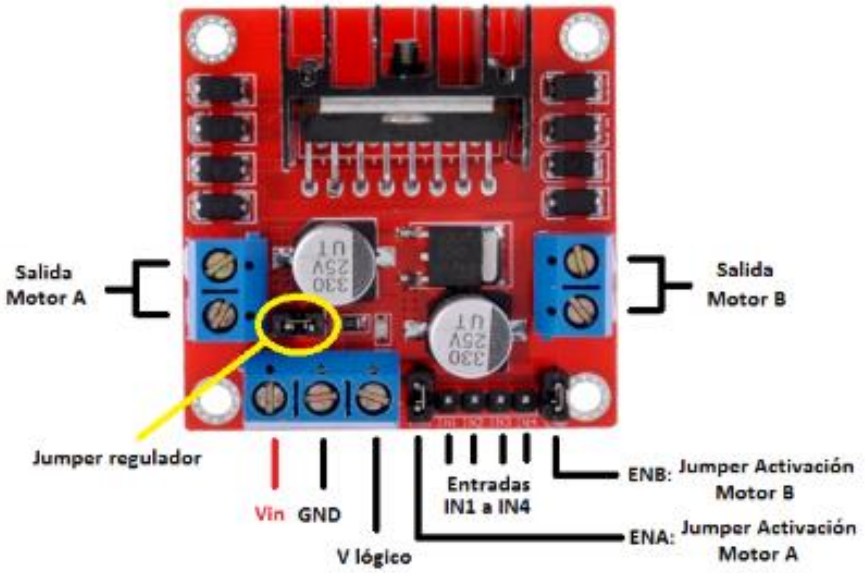


Figura 31. Driver L298N

La entrada de tensión Vin admite tensiones entre 3V y 35V y justo a su derecha en la imagen está el pin que se debe conectar a GND. La conexión V lógico puede funcionar de dos maneras distintas. Si el jumper del regulador está cerrado se activa el regulador de tensión del módulo y en V lógico se tiene una

salida de 5V que puede usarse por ejemplo para alimentar un Arduino. Por otra parte, si se quita el jumper se desactiva el regulador y por tanto es necesario alimentar la parte lógica del módulo, así que hay que introducir una tensión de 5V por la conexión V lógico para que el módulo funcione. Si se introduce una corriente por V lógico con el jumper de regulación puesto se podría dañar el módulo.

Las salidas para los motores A y B proporcionan la energía necesaria para mover los motores. Los pines IN1 e IN2 sirven para controlar el sentido de giro del motor A, y los pines IN3 e IN4 el del motor B. Para poder controlar la velocidad de giro de los motores hay que quitar los jumpers y utilizar los pines ENA y ENB. Se conectan a dos salidas PWM de Arduino de manera que se le envía un valor entre 0 y 255 que controle la velocidad de giro.

7.2.3.1 PWM (Pulse Width Modulation)

La modulación por ancho de pulsos PWM es una técnica que se basa en modificar el ciclo de trabajo de una señal periódica (normalmente suele ser una señal cuadrada) para transmitir información o para controlar la cantidad de energía que se envía a los motores como ocurre en este trabajo. Por lo tanto, esta técnica permite regular tanto la velocidad como el par de los motores de corriente continua. En la etapa de control se genera una señal PWM que se envía al driver L298N y de esta forma el driver regula la tensión que llega a los bornes de los motores y se controla la velocidad de estos.

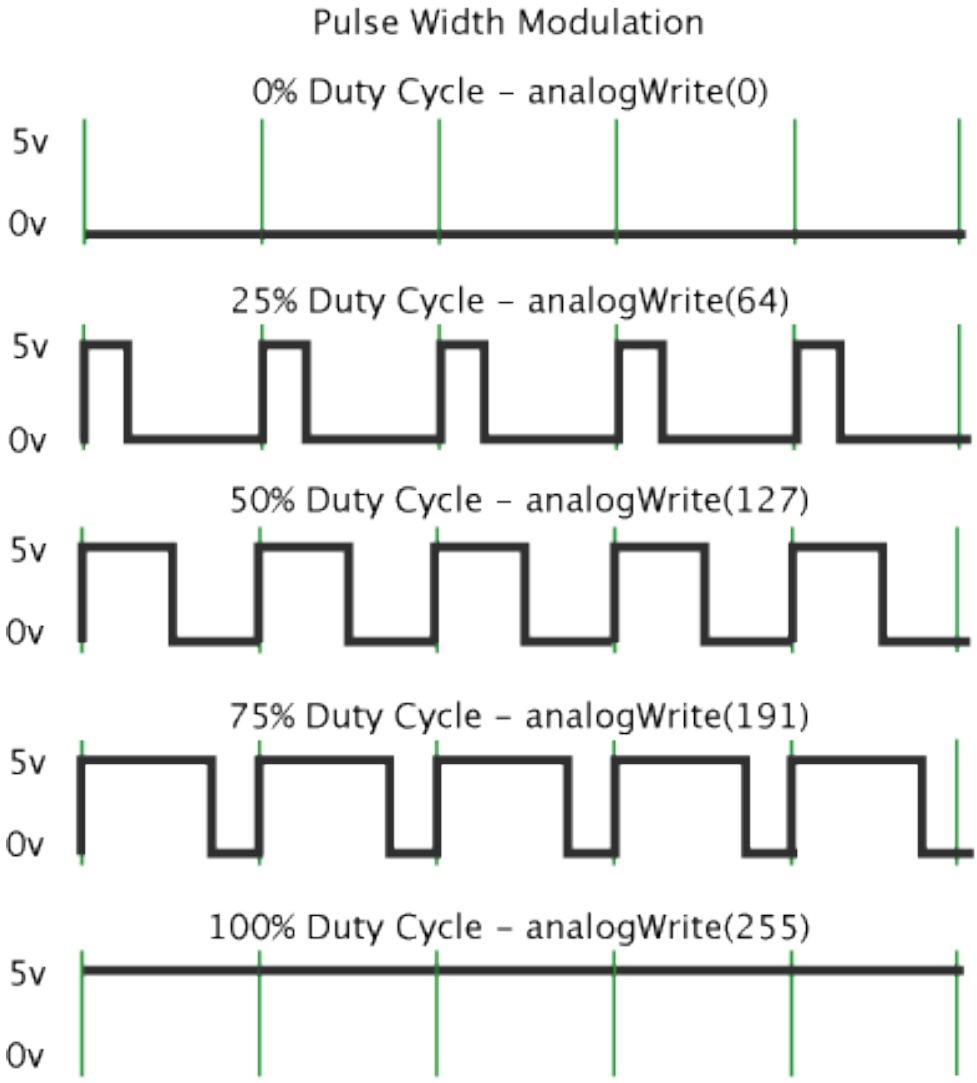


Figura 32. Señales PWM

7.2.3.2 PUENTE EN H

El puente en H es un circuito electrónico que permite cambiar el sentido de giro de los motores de corriente continua. Dicho circuito lo componen cuatro interruptores, los cuales pueden ser mecánicos o electrónicos (diodos, transistores…) de manera que, según cuales se abran o se cierren, permitirán el cambio de sentido de la alimentación de un motor. Se pueden encontrar como circuitos integrados y también pueden ser creados con componentes discretos.

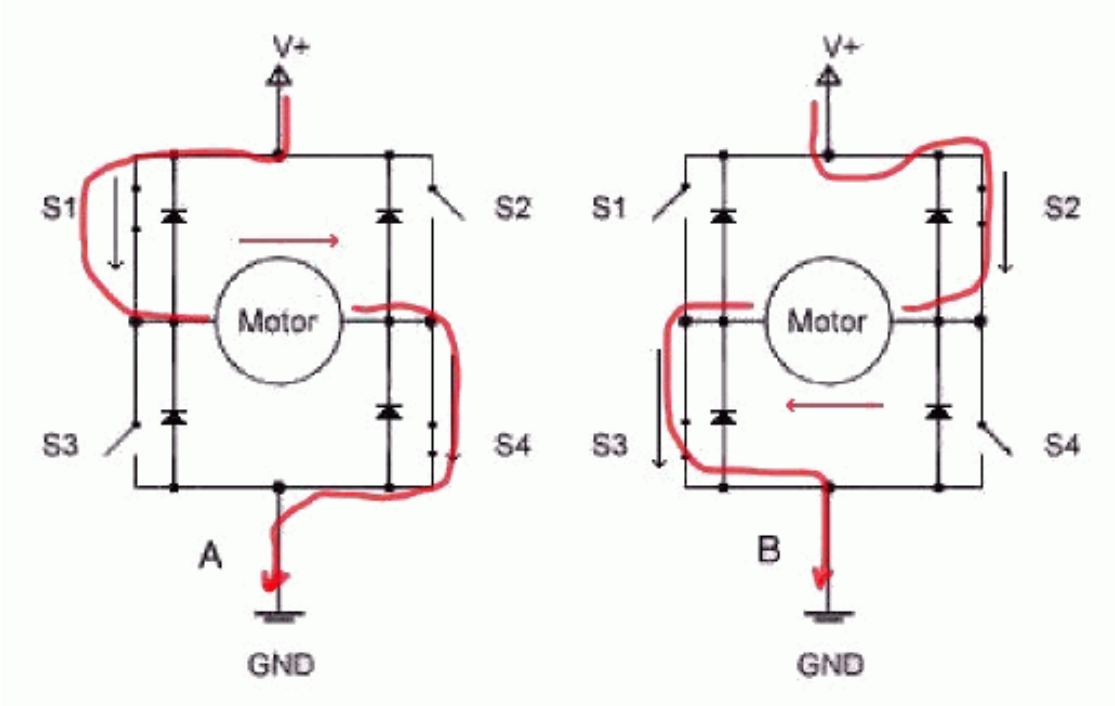


Figura 33. Puente en H

El funcionamiento de un puente en H es bastante sencillo. Cuando se cierran los interruptores S1 y S4, y se abren S2 y S3, se aplica una tensión positiva en los bornes del motor, por lo tanto, este gira en un sentido. Por el contrario, cuando se cierran los interruptores S2 y S3 y se abren S1 y S4, se invierte la tensión en los bornes del motor por lo que este gira en sentido contrario. Si se da el caso de que todos los interruptores estén abiertos el motor no girará.

7.2.3.3 ELECCIÓN DEL DRIVER L298N

Para el driver de continua que controla a los dos motores se han mirado dos modelos distintos, el L293D y el L298N.

Son dos drivers con características muy similares. El L293D tiene cuatro canales mientras que el L298N tiene dos, los suficientes para los dos motores. Al L293D le sobran dos canales que podrían ser usados para controlar relés, solenoides, lámparas… Los voltajes que admiten son parecidos, unos 35V de voltaje máximo. Sin embargo, las intensidades varían, la corriente de salida por canal en el L293D es de 1A mientras que en el L298N es del doble, 2A.

Económicamente son similares también de precio así que finalmente se ha optado para utilizar en este proyecto el driver L298N puesto que la corriente que aporta a los motores es mayor teniendo en cuenta que las demás características importantes son prácticamente las mismas.

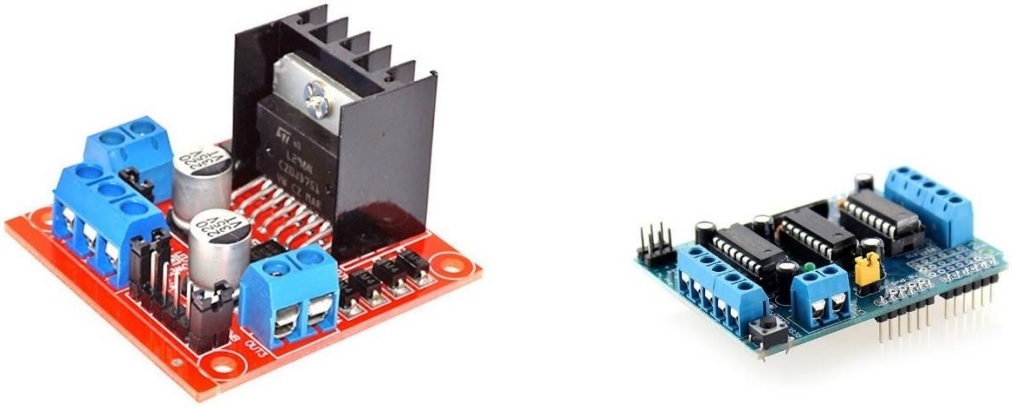


Figura 34. A la izda. driver L298N y a la dcha. driver L293D

7.2.4 ARDUINO

7.2.4.1 INTRODUCCIÓN

Arduino es una plataforma de electrónica de código abierto basada en software y hardware fácil de usar, pensada para que se puedan crear proyectos sin la necesidad de elaborar PCBs complejas y permitiendo el prototipado rápido de la solución deseada. Esta plataforma se basa en combinar dos soluciones diferentes. Por un lado, el hardware está basado en diferentes placas de desarrollo de muy bajo coste basadas en microcontroladores ATMEGAxxx. Estos microcontroladores son de la familia RISC de 8 bits de la empresa AMTEL con arquitectura Harvard y son el principal competidor de los PIC de 8 bits. Por otro lado, existe un IDE gratuito que, usando librerías de funciones muy sencillas, permite desarrollar software en C para este tipo de microcontroladores. Arduino se ha usado en muchos proyectos electrónicos tanto a nivel de estudiantes como en complejos instrumentos científicos, debido a la facilidad de acceso tanto al hardware como al software. El ser de código abierto y su gran difusión han provocado una gran comunidad mundial de desarrolladores y su uso en multitud de proyectos diversos. Por todo esto, es muy sencillo encontrar soluciones a los errores que se puedan detectar, lo que facilita bastante el aprendizaje tanto de electrónica como de programación.

Figura 35. Logo de Arduino

7.2.4.2 ENTORNO DE DESARROLLO

El entorno de desarrollo es muy sencillo ya que el programa se escribe en la zona del sketch y para compilarlo y cargarlo a cualquier placa solo se necesita pulsar un botón.

El software de desarrollo de Arduino se publica bajo una licencia libre y gratuita que se puede obtener de la página oficial de Arduino e incluye los drivers necesarios para cualquier placa Arduino, así como todas las librerías necesarias para su programación. Sin embargo, uno de los principales inconvenientes que presenta el IDE es que no incorpora un debugger, complicándose de esta manera la detección de errores en proyectos complejos.

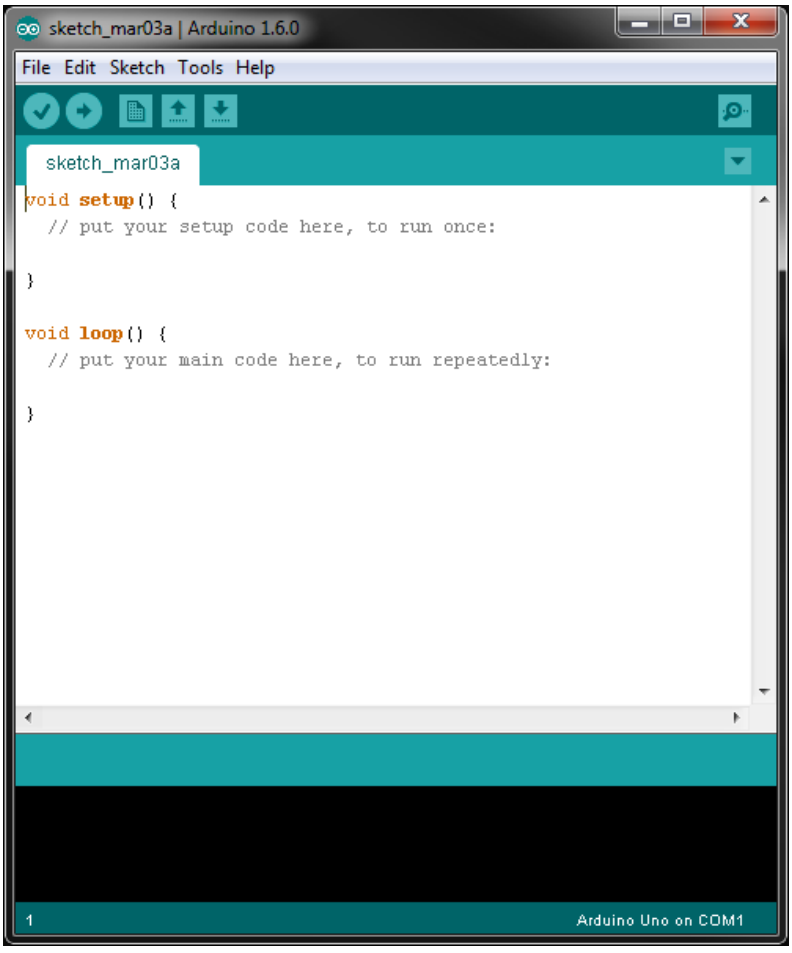


Figura 36. IDE de Arduino

7.2.4.3 VENTAJAS DE ARDUINO

Arduino presenta una serie de ventajas que son clave para elegirlo a la hora de realizar un proyecto y así no tener que usar un PIC como microcontrolador:

- El software de desarrollo de Arduino está publicado bajo una licencia libre por lo que es totalmente gratuito. Desde la página web oficial de Arduino se puede descargar tanto el programa como los drivers y librerías que podamos necesitar también de forma totalmente gratuita.
- Hay multitud de documentación disponible, tanto en libros como en Internet, donde se puede encontrar librerías completas para el uso de diferentes dispositivos que se le pueden conectar, diversos ejemplos de programas… Además, todas estas librerías son también de código abierto.
- Una placa de Arduino lleva ya incorporado el microcontrolador y todos los elementos necesarios para su funcionamiento y programación. Si por el contrario se utiliza un PIC, sería necesario diseñar una PCB y fabricarla, lo cual conlleva tener muchas posibilidades de que se cometa un error.
- Cuando se va a cargar un programa en la tarjeta Arduino, esta se conecta al ordenador vía serie mediante un puerto USB. Sin embargo, con los PIC es necesaria una tarjeta de programación para realizar dicha operación.
- Se puede conectar de manera sencilla un dispositivo externo a la tarjeta como pudiera ser un teclado, una pantalla LCD, un sensor… y con una programación simplificada gracias a las librerías y ejemplos gratuitos que hay disponibles en la web.

Por todas estas razones se ha elegido usar Arduino en vez de otro microcontrolador.

7.2.4.4 PLACAS DE ARDUINO

Como ya se ha comentado anteriormente, existen diferentes placas de desarrollo disponibles y las más importantes son las siguientes:

- Arduino Uno R3: está basada en Atmega328P la cual tiene 14 entradas/salidas digitales (6 de las cuales pueden ser utilizadas como salidas PWM), 6 entradas analógicas y un reloj de 16 MHz.

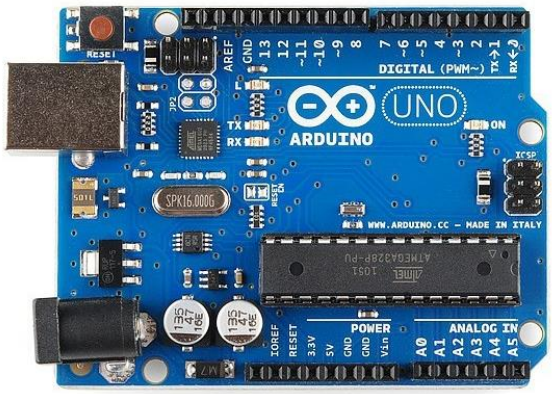


Figura 37. Arduino Uno R34

- Arduino Mega: es la versión expandida del Arduino Uno y tiene un procesador ATmega2560. Dispone de 54 entradas/salidas digitales (de las cuales 15 se pueden usar como salidas PWM), 16 entradas analógicas y 4 UART (hardware serial ports).

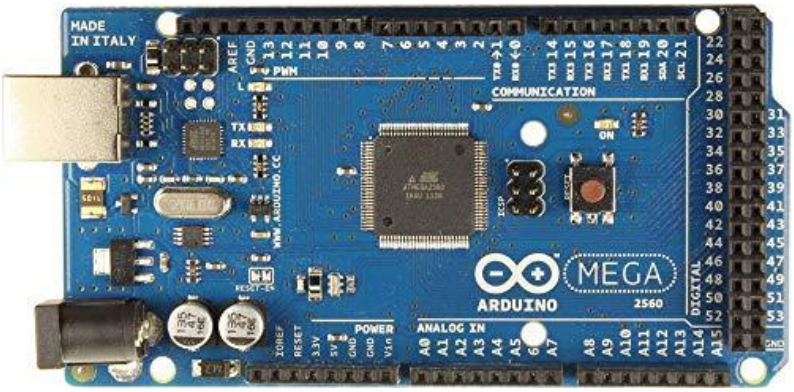


Figura 38. Arduino Mega

- Arduino Nano: es parecido al Arduino Uno, pero con un empaquetado mucho más reducido. Está basado también en Atmega328P, pero este no

dispone de un regulador de tensión de 5V y la conexión con el ordenador se realiza mediante un USB del tipo Mini-B.

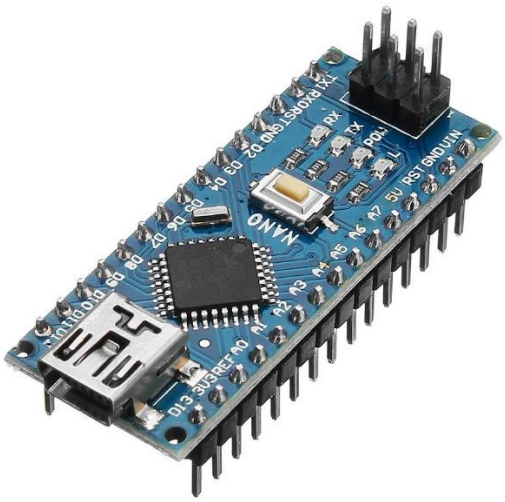


Figura 39. Arduino Nano

- Arduino Yún LininoOS: está basado en Atmega32u4 y en Atheros AR9331. Esta placa soporta una distribución Linux basada en OpenWRT llamado Linino OS, ofreciendo soporte Ethernet, Wifi, USB y micro SD.

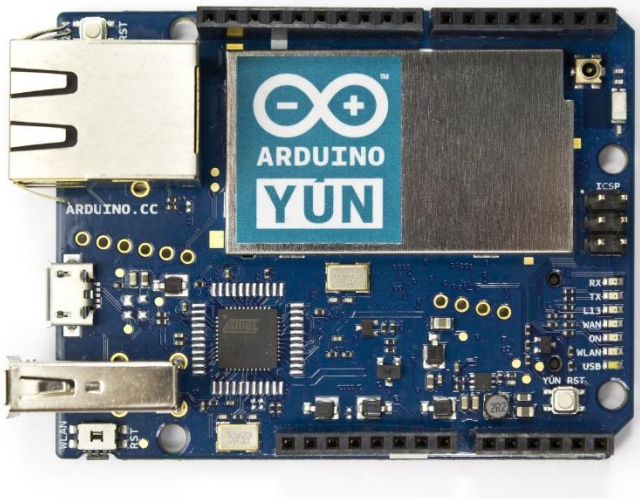


Figura 40. Arduino Yún LininoOS

Finalmente, de todas estas placas de Arduino se ha decidido utilizar en este proyecto la de Arduino Uno. Es ligera de peso y satisface ampliamente las necesidades del sistema. Además, económicamente tiene una relación calidad/precio muy buena.

7.2.5 ENCODERS

El uso de encoders en este proyecto es una opción optativa puesto que no son necesarios para mantener el equilibrio del robot. Para lo que se van a emplear principalmente es para calcular la velocidad angular de las ruedas del robot y así poder hallar las constantes necesarias en el modelo matemático.

Los encoders son dispositivos electromecánicos que permiten codificar el movimiento mecánico en distintos tipos de impulsos eléctricos: analógicos en función de una onda, digitales binarios, pulsos… De esta manera, los encoders son una interfaz entre un dispositivo mecánico móvil y un controlador.

Existen dos grandes tipos de encoders: los lineales y los rotatorios. En cada grupo hay además distintos tipos de codificación y diversos principios electromecánicos de funcionamiento que se van a explicar en los anexos.

Para este proyecto se ha utilizado un tipo de encoder rotatorio que tiene una resolución de 20 pulsos por revolución, suficiente para poder calcular de manera óptima las rpm de los motores.

Figura 41. Motor DC con encoder rotatorio

7.2.6 ALIMENTACIÓN DEL SISTEMA

En primer lugar, se utilizó una batería recargable de 9V para alimentar la etapa de potencia y la de control. Estas baterías tienen la ventaja de ser fáciles de usar y encontrar. Además, hay disponibles cables y porta pilas. Como inconvenientes las baterías de 9V disponen de baja densidad energética. Una pila tiene una capacidad de 500-600mAh. Además, proporcionan una intensidad de corriente máxima muy baja, unos 300mA. Para un proyecto como este podrían servir perfectamente pero aun así se ha optado por una opción que proporcione una mayor intensidad y autonomía al sistema.

Figura 42. Batería recargable de 9V

La opción elegida son seis pilas AA de 1,5V cada una, en total 9V, la misma tensión que aportaba la batería recargable. Lo que varía es la carga que es superior a las pilas de 9V. Seis pilas AA convencionales proporcionan 1200-1900 mAh, más del doble de lo que aportaba la batería de 9V. La intensidad máxima que se puede obtener supera 1A, pudiendo llegar a extraerse hasta 2A. Hay que tener en cuenta también que la cantidad de carga que podemos extraer de la pila se reduce cuanto más rápido la drenamos. El principal inconveniente que tienen es que no son recargables. Por lo demás sirven perfectamente para nuestro proyecto.

Figura 43. De izda. a dcha. pilas AA de 1,5V cada una y portapilas

8 MATLAB Y SIMULINK

Matlab es un sistema de cálculo numérico que ofrece un entorno de desarrollo integrado (IDE) provisto de un lenguaje de programación propio (lenguaje M). Está disponible para las plataformas Unix, Mac OS X, Windows y GNU/Linux.

Entre sus principales prestaciones se encuentran: la representación de datos y funciones, la manipulación de matrices, la implementación de algoritmos, la comunicación con programas en otros lenguajes y con otros dispositivos hardware y la creación de interfaces de usuario. El paquete Matlab se compone de dos herramientas adicionales que expanden sus prestaciones como son Simulink (plataforma de simulación) y Guide (editor de interfaces de usuario). Además, las capacidades de Matlab se pueden ampliar con las toolboxes y las de Simulink con los paquetes de bloques.

Matlab es un software muy utilizado en universidades y centros de investigación y desarrollo. En los últimos años el número de prestaciones se ha visto incrementado ya que se puede programar directamente procesadores digitales de señal o se puede crear código VHDL.

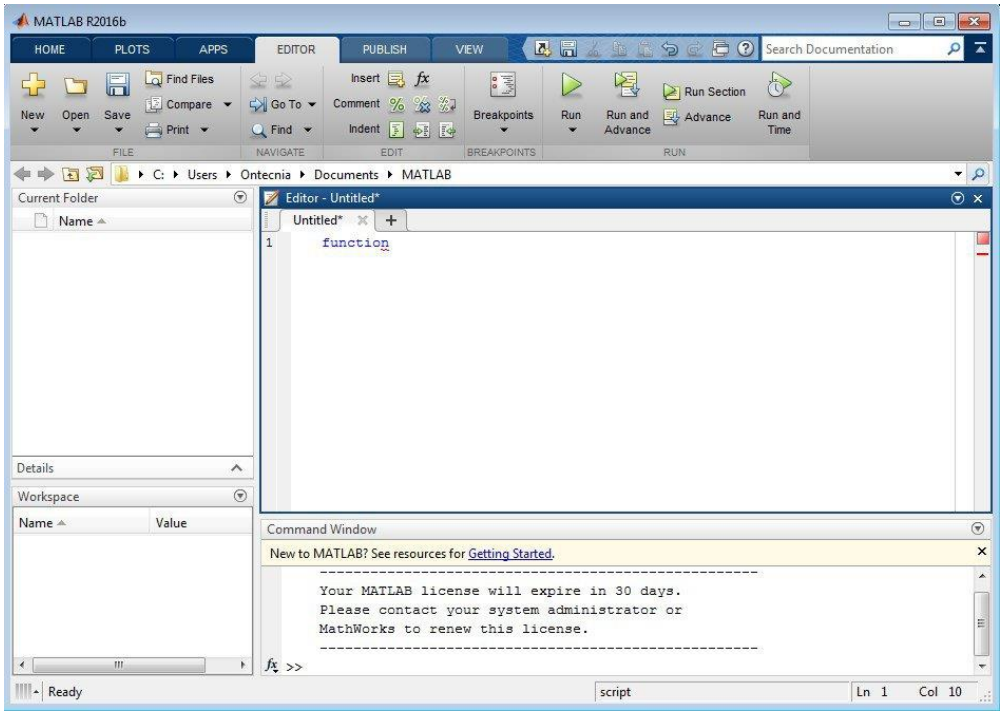


Figura 44. Entorno de desarrollo de Matlab

Figura 45. Logo de Matlab y Simulink5

Simulink es un entorno de programación visual que trabaja sobre el entorno de programación de Matlab. Tiene un nivel de abstracción más alto que el lenguaje interpretado en Matlab y genera archivos con extensión .mdl.

Simulink es una herramienta de simulación de modelos o sistemas, con cierto grado de abstracción de los fenómenos físicos involucrados en los mismos.

Es bastante utilizado en ingeniería electrónica en temas relacionados con el procesamiento digital de señales, involucrando temas específicos de telecomunicaciones, ingeniería biomédica… También es muy utilizado en ingeniería de control y robótica.

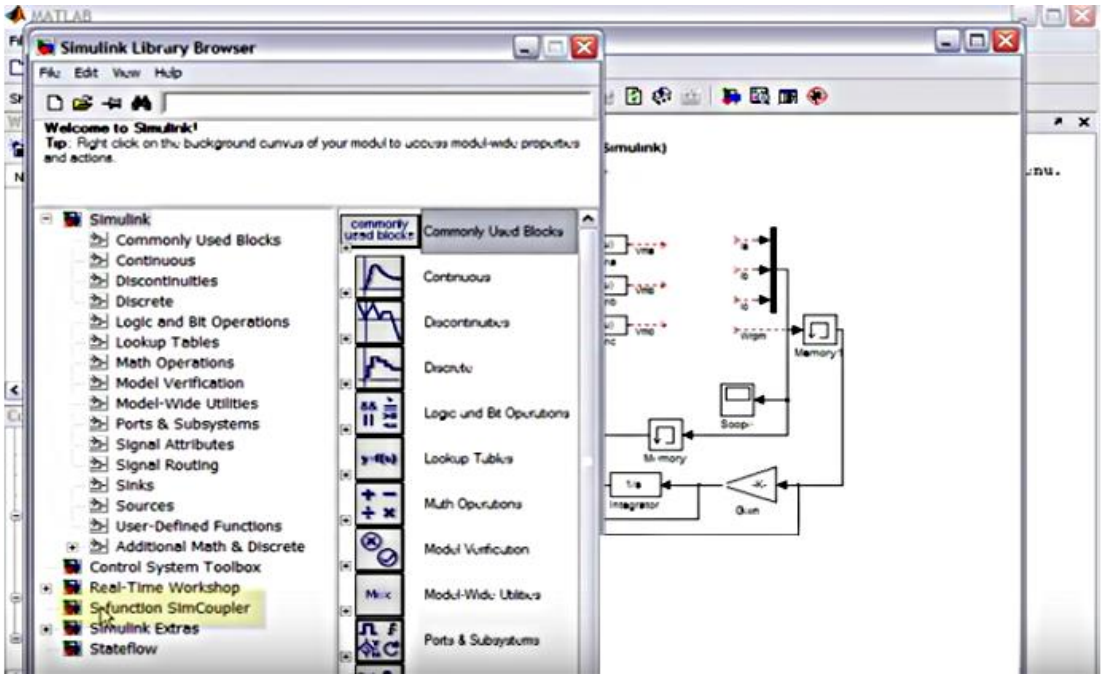


Figura 46. Entorno de desarrollo de Simulink

9 MODELO MATEMÁTICO DEL SISTEMA

A continuación, se va a analizar el problema físico del equilibrio de un péndulo invertido que es un sistema con una inestabilidad estática la cual puede ser compensada aplicando un momento o un par determinado en el eje de giro.

La base de este trabajo consiste en tratar de conseguir controlar dicho equilibrio teniendo en cuenta para ello el ángulo formado con la vertical y las aceleraciones que actúan en el robot.

El estudio de estos sistemas inestables constituye uno de los problemas clásicos analizados en la teoría de control y en la dinámica de sistemas, y por ello es tratado en numerosos libros. También este problema suele ser empleado para trabajar con diferentes tipos de controles como pueden ser el PID, los sistemas de control en el espacio de estados…

Si se considera el problema como un péndulo invertido y se aproxima como punto de balanceo fijo y sistema de movimiento en dos dimensiones, se obtiene la siguiente ecuación de comportamiento de nuestro sistema:

$$\ddot{\emptyset} = \frac{g}{L} \, sen \, \emptyset$$

Como se puede ver la aceleración angular ($\ddot{\emptyset}$) depende de la gravedad (g), la longitud del péndulo (L) y el seno del ángulo formado con la vertical ($\emptyset$).

En los apartados posteriores se presenta el modelo matemático completo del robot equilibrista. En la siguiente imagen se muestra la vista lateral y el sistema de coordenadas sobre el cual se realiza el modelo matemático para el péndulo invertido sobre dos ruedas.

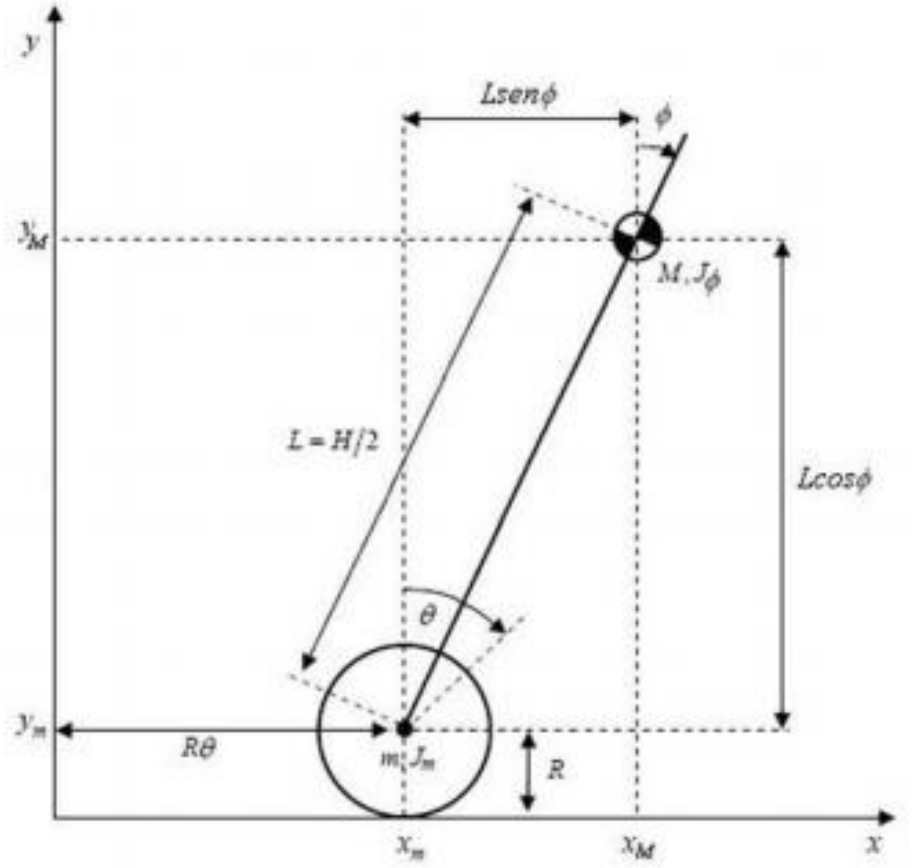


Figura 47. Modelo del péndulo invertido

La descripción matemática del robot equilibrista se divide en tres partes, una sería la estructura del robot que tiene un comportamiento similar al de un péndulo invertido, otra parte serían las ruedas del robot y la última sería el sistema eléctrico de los motores de continua. El péndulo invertido y las ruedas tienen tres ecuaciones cada una, una para la dirección rotacional y dos para las direcciones en el eje X y en el eje Y.

El ángulo de giro para el péndulo se ha considerado ∅ y para las ruedas θ.

9.1 MODELO MATEMÁTICO DE LA ESTRUCTURA DEL ROBOT

En la siguiente imagen se pueden observar las fuerzas que actúan sobre la estructura del robot:

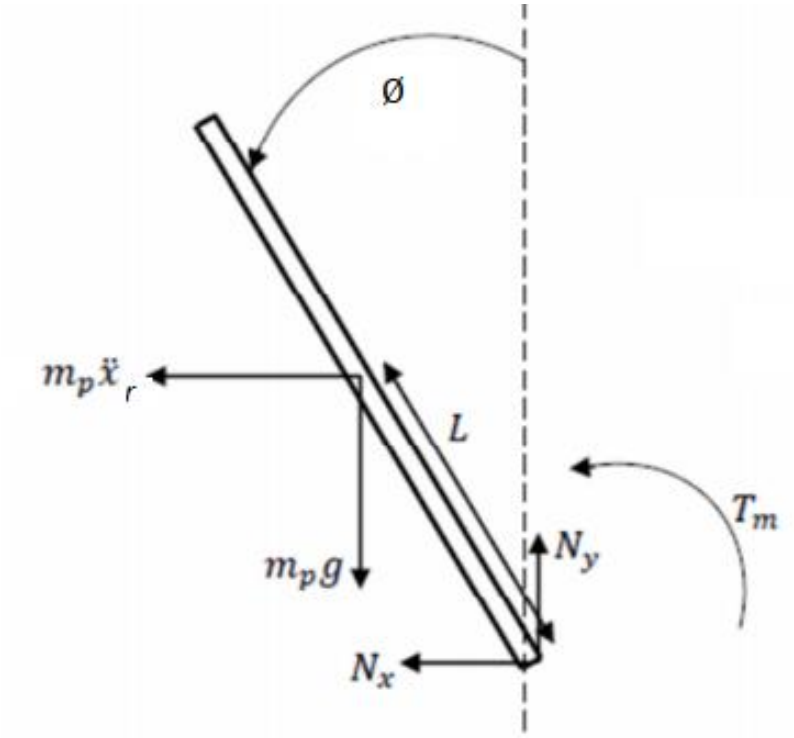


Figura 48. Modelo de la estructura del robot

La estructura del robot se modela como un péndulo invertido con las ecuaciones que se muestran a continuación.

La siguiente ecuación es la suma de momentos angulares respecto al eje de giro del péndulo:

$$\sum M_0 = I\alpha \quad [Nm]$$

$$T_m + m_p g L sen\emptyset + m_p \ddot{x}_r L cos\emptyset = J_p \ddot{\emptyset}$$

- T_m es el par de los motores
- $m_p g L sen\emptyset$ es el momento creado por la aceleración de la gravedad
- $m_p \ddot{x}_r L cos\emptyset$ es el momento generado por la aceleración de las ruedas
- $J_p \ddot{\emptyset}$ es la inercia del péndulo por su aceleración angular

Las siguientes ecuaciones son las fuerzas que actúan en la dirección X e Y respectivamente:

$$\sum F = ma \quad [N]$$

$$-N_x - m_p\ddot{x}_r = m_p\ddot{x}_p$$

$$N_y - m_p g = m_p\ddot{y}_p$$

A partir de estas ecuaciones se obtienen las fuerzas reactivas en el eje X y en el eje Y respectivamente:

$$N_x = m_p(-\ddot{x}_r + L\ddot{\emptyset}cos\emptyset - L\dot{\emptyset}^2 sen\emptyset)$$

$$N_y = m_p(g - L\ddot{\emptyset}sen\emptyset - L\dot{\emptyset}^2 cos\emptyset)$$

Las aceleraciones $\ddot{x}_p$ y $\ddot{y}_p$ se trasladan desde las coordenadas X-Y del sistema hasta la coordenada rotacional:

$$x_p = -Lsen\emptyset$$

$$y_p = Lcos\emptyset$$

Las ecuaciones anteriores se derivan para obtener las velocidades:

$$\dot{x}_p = -L\dot{\emptyset}cos\emptyset$$

$$\dot{y}_p = -L\dot{\emptyset}sen\emptyset$$

Y estas se vuelven a derivar para obtener las aceleraciones:

$$\ddot{x}_p = -L\ddot{\emptyset}cos\emptyset + L\dot{\emptyset}^2 sen\emptyset$$

$$\ddot{y}_p = -L\ddot{\emptyset}sen\emptyset - L\dot{\emptyset}^2 cos\emptyset$$

Para calcular la inercia de la estructura del robot, esta se ha considerado como un prisma rectangular con una distribución de masa uniforme. Primero se calcula la inercia respecto al eje paralelo al de las ruedas pero que pasa por el centro de masas y luego se traslada al eje de las ruedas que es el eje de rotación aplicando el Teorema de Steiner.

$$J_p = \frac{1}{12}m_p H^2 + \frac{1}{12}m_p w^2 + m_p L^2 \quad [kgm^2]$$

9.2 MODELO MATEMÁTICO DE LAS RUEDAS

En la siguiente imagen se pueden observar las fuerzas que actúan sobre una rueda del robot:

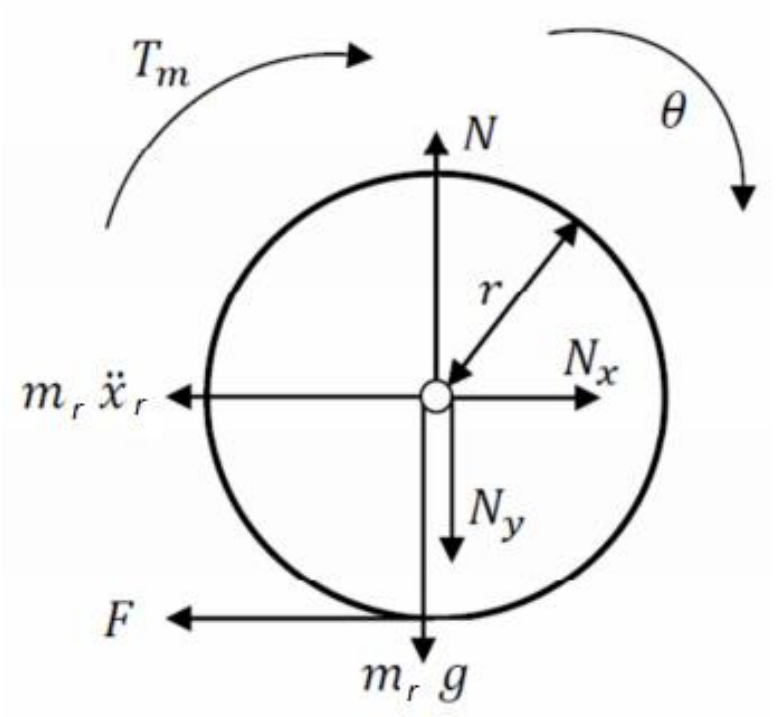


Figura 49. Modelo de las ruedas

Las ecuaciones que describen este modelo son las citadas a continuación.

La siguiente ecuación es la suma de momentos angulares respecto al eje central de las ruedas que es el eje de giro:

$$\sum M_0 = I\alpha \quad [Nm]$$

$$T_m + Fr = J_r\ddot{\theta}$$

Las ecuaciones siguientes son las fuerzas que actúan en las direcciones X e Y respectivamente:

$$\sum F = ma \quad [N]$$

$$N_x - F = m_r\ddot{x}_r$$

$$N - N_y - m_r g = m_r\ddot{y}_r$$

Para trasladar las aceleraciones $\ddot{x}_r$ y $\ddot{y}_r$ desde el sistema de coordenadas X-Y al sistema de coordenadas rotacional se utilizan las siguientes ecuaciones:

$$\ddot{x}_r = \ddot{\theta}_r$$

$$\ddot{y}_r = 0$$

Para calcular la inercia de las ruedas se utiliza la fórmula del círculo respecto a un eje perpendicular a él que pase por el centro:

$$J_r = \frac{1}{2} m_r r^2 \quad [kgm^2]$$

9.3 MODELO MATEMÁTICO DE LOS MOTORES DE CONTINUA

Para tratar de estabilizar el robot se emplean dos pequeños motores de corriente continua. Se va a realizar una descripción matemática de dichos motores para encontrar una relación entre el voltaje de entrada, la señal de control y el par entregado desde los motores.

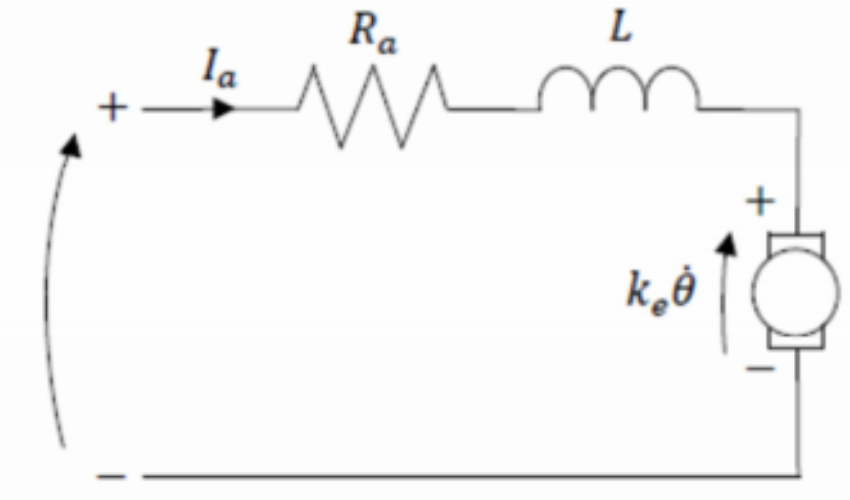


Figura 50. Esquema eléctrico de los motores

La ecuación que describe el comportamiento del motor es la siguiente:

$$U = R_a I_a + K_e \dot{\theta} + L \frac{di}{dt} \quad [V]$$

Como el tiempo de la constante eléctrica es mucho más pequeño que el de la constante mecánica, la inductancia puede ser despreciada quedando lo siguiente:

$$U = R_a I_a + K_e \dot{\theta}$$

La siguiente ecuación muestra el par del eje del motor:

$$T_m = n K_t I_a \quad [Nm]$$

El par en el eje del motor tiene que superar la inercia del motor, el amortiguamiento viscoso y la fricción del motor. Todos ellos son valores muy pequeños difíciles de estimar por tanto la fricción y el amortiguamiento viscoso se pueden despreciar. De esta manera queda la ecuación anteriormente descrita.

Las constantes del motor K_t y K_e tienen que ser también calculadas, al igual que la resistencia de armadura del motor R_a. Para obtener dichos valores se ha realizado un experimento en el laboratorio que se explicará en apartados posteriores.

Como los motores tienen que ser controlados a través de voltaje controlado PWM, la corriente se elimina de la ecuación de par quedando la ecuación de la siguiente manera:

$$I_a = \frac{U - K_e\dot{\theta}}{R_a}$$

$$T_m = \frac{nK_tU}{R_a} - \frac{nK_tK_e\dot{\theta}}{R_a}$$

9.4 EXPERIMENTO PARA CALCULAR LAS CONSTANTES DEL MOTOR Y LA RESISTENCIA DE ARMADURA

Se quieren obtener los parámetros necesarios del motor de continua para poder calcular de manera correcta la función de transferencia del sistema global. Estos datos no aparecen en las especificaciones del motor y por tanto se ha realizado un experimento en el laboratorio para poder estimar dichos valores.

En un motor de continua aparece en el inducido una tensión proporcional al producto del flujo por la velocidad angular, cuando dicho motor se encuentra en rotación. Si el flujo es constante como ocurre en este caso, la tensión inducida E_b es directamente proporcional a la velocidad angular.

$$K_e = \frac{E_b}{\omega}$$

- K_e es la constante contraelectromotriz del motor
- E_b es la fuerza contraelectromotriz [V]
- ω es la velocidad del motor en rad/s

Para calcular la fuerza contraelectromotriz se emplea la siguiente fórmula:

$$E_b = V - IR_a$$

R_a es la resistencia de armadura del motor y se calcula midiendo con un multímetro la resistencia en los devanados de la armadura del motor. $R_a = 7\Omega$

El voltaje se aplica mediante la fuente de alimentación regulable del laboratorio y la intensidad se mide con el amperímetro.

En la siguiente tabla aparecen los datos medidos experimentalmente necesarios para poder calcular K_e a partir de cinco valores de voltaje de entrada diferentes.

Voltaje [V]	Intensidad [A]	E_b [V]	n [rpm]	ω [rad/s]	K_e [V/rad/s]
3	0.06	2.58	320	33.51	0.077
4	0.07	3.51	420	43.98	0.0798

5	0.08	4.44	530	55.5	0.08
6	0.09	5.37	625	65.45	0.082
7	0.1	6.3	725	75.92	0.083

Tabla 1. Datos para obtener K_e

Para hallar las rpm del motor de continua se ha creado un programa en Arduino que mide las rpm en función de los pulsos que genera el encoder.

```
//Variables globales

int rpm;

volatile byte pulsos;

unsigned long timeold;

unsigned int pulsos_por_vuelta = 20;


//Interrupcion

void contador ()

{

  pulsos++;

}


void setup() {

  Serial.begin(115200);

  pinMode(2,INPUT);


  attachInterrupt(0,contador,RISING);

  pulsos = 0;

  rpm = 0;

  timeold = 0;

}
```

```
void loop() {
  if (millis() - timeold >= 1000)
  {
    detachInterrupt(0);
    rpm = (60 * 1000 / pulsos_por_vuelta)/(millis() - timeold) * pulsos;
    timeold = millis();
    pulsos = 0;

    Serial.print("RPM = ");
    Serial.println(rpm,DEC);

    attachInterrupt(0,contador,RISING);
  }
}
```

Figura 51. Programa de Arduino para calcular las rpm

En la siguiente imagen se puede ver el experimento realizado en el laboratorio para poder calcular dichas rpm. Con la fuente de alimentación se va variando el voltaje de entrada y con el encoder conectado a la placa de Arduino se van midiendo las rpm del motor de continua.

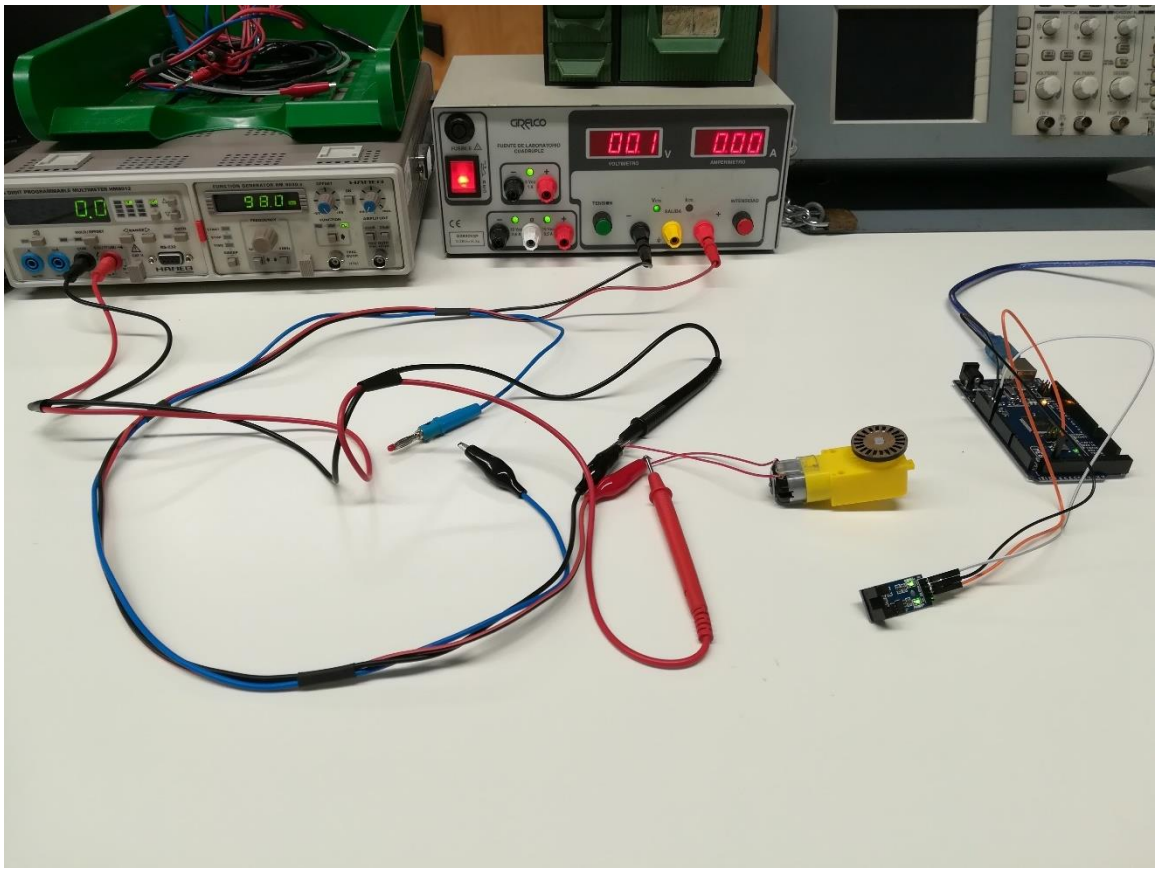


Figura 52. Experimento para hallar K_e

Una vez que se tienen todos los datos calculados, se hace la media de los valores de K_e para calcular el valor de la constante contraelectromotriz de una manera más precisa:

$$K_e = \frac{0.077 + 0.0798 + 0.08 + 0.082 + 0.083}{5} = 0.08 \frac{V}{rad/s}$$

Para calcular el valor de K_t se emplea la técnica llamada paramétrico dimensional que no recurre a la prueba experimental, pero está comprobado que es muy útil y confiable para los motores de continua. Se basa en utilizar expresiones que guardan una relación paramétrica dimensional directa entre K_t y la constante contraelectromotriz K_e y cuya comprobación empírica ha sido sustentada.

$$K_t \left(\frac{Nm}{A}\right) = K_e \left(\frac{V}{\frac{rad}{s}}\right) = 0.08$$

9.5 MODELO LINEALIZADO MEDIANTE ESPACIO DE ESTADOS

A partir de las ecuaciones de los tres modelos anteriormente descritos, el modelo de estado queda definido de la siguiente forma:

Las variables de estado del sistema son las siguientes:

$$X = \begin{bmatrix} \emptyset \\ \dot{\theta} \\ \dot{\emptyset} \end{bmatrix}$$

Donde:

- $\emptyset$ es el ángulo del péndulo
- $\dot{\theta}$ es la velocidad angular de las ruedas
- $\dot{\emptyset}$ es la velocidad angular del péndulo

El modelo de espacio de estados va a tener por tanto tres estados:

$$\dot{X} = AX + BU$$

$$Y = CX + DU$$

Donde:

$$A = \begin{bmatrix} 0 & 0 & 1 \\ a_1 & a_2 & 0 \\ a_3 & a_4 & 0 \end{bmatrix}$$

$$B = \begin{bmatrix} 0 \\ b_1 \\ b_2 \end{bmatrix}$$

$$C = [1 \quad 0 \quad 0]$$

$$D = [0]$$

$$\dot{X} = \begin{bmatrix} \dot{\emptyset} \\ \ddot{\theta} \\ \ddot{\emptyset} \end{bmatrix}$$

Despejando las variables de estado del sistema en las ecuaciones de los modelos anteriormente citados se obtienen los coeficientes de las matrices A y B:

$$a_1 = \frac{gL^2 r {m_p}^2}{-L^2 r^2 {m_p}^2 + J_p(J_r + r^2(m_p + m_r))}$$

$$a_2 = \frac{-2nJ_p K_e K_t - 2nLrK_e K_t m_p}{(-L^2 r^2 {m_p}^2 + J_p\left(J_r + r^2\left(m_p + m_r\right)\right))R_a}$$

$$a_3 = \frac{gLJ_r m_p R_a + Lm_p r^2(gm_p + gm_r)R_a}{(-L^2 r^2 {m_p}^2 + J_p\left(J_r + r^2\left(m_p + m_r\right)\right))R_a}$$

$$a_4 = \frac{-2nJ_r K_e K_t - 2nrK_e K_t((L + r)m_p + rm_r)}{(-L^2 r^2 {m_p}^2 + J_p\left(J_r + r^2\left(m_p + m_r\right)\right))R_a}$$

$$b_1 = \frac{2nJ_p K_t + 2nLrK_t m_p}{(-L^2 r^2 {m_p}^2 + J_p\left(J_r + r^2\left(m_p + m_r\right)\right))R_a}$$

$$b_2 = \frac{2nJ_r K_t + 2nrK_t((L + r)m_p + rm_r)}{(-L^2 r^2 {m_p}^2 + J_p\left(J_r + r^2\left(m_p + m_r\right)\right))R_a}$$

9.6 OBTENCIÓN DE LA FUNCIÓN DE TRANSFERENCIA

La función de transferencia de un modelo de espacio de estados continuo e invariante en el tiempo se puede obtener de la siguiente forma:

Calculando la transformada de Laplace de la siguiente ecuación:

$$\dot{x}(t) = Ax(t) + Bu(t)$$

Se obtiene:

$$sX(s) = AX(s) + BU(s)$$

Después, factorizando $X(s)$:

$$sX(s) - AX(s) = BU(s)$$

$$(sI - A)X(s) = BU(s)$$

I es la matriz identidad:

$$I = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

Despejando $X(s)$:

$$X(s) = (sI - A)^{-1}BU(s)$$

Sustituyendo $X(s)$ en la ecuación de salida:

$$Y(s) = CX(s) + DU(s)$$

Queda lo siguiente:

$$Y(s) = C((sI - A)^{-1}BU(s)) + DU(s)$$

La función de transferencia de un sistema es la relación entre la salida y la entrada:

$$G(s) = \frac{Y(s)}{U(s)}$$

Sustituyendo las ecuaciones anteriores por $Y(s)$ con respecto a $U(s)$, se obtiene la expresión final:

$$G(s) = C(sI - A)^{-1}B + D$$

Para no tener que realizar todos esos cálculos de la expresión anterior a mano se utiliza un script de Matlab para calcular la función de transferencia de una forma más sencilla y eficiente.

```
%% Variables

g = 9.81;    %gravedad
H = 0.16;    %altura del pendulo
w = 0.08;    %anchura del pendulo
L = 0.07;    %distancia al centro de masas
r = 0.03;    %radio de la rueda
mp = 0.548;    %masa del pendulo
mr = 0.083;    %masa de las ruedas
Jp = 1/12*mp*H^2+1/12*mp*w^2+mp*L^2;    %inercia del pendulo
Jr = 1/2*mr*r^2;    %inercia de las ruedas
Ra = 7;    %resistencia de armadura
Kt = 0.08;  %kt=ke
Ke = 0.08;  %ke=fem/w
n = 1;    %relacion de transmisión
```

```
%% Ecuaciones

a1 = (g*L^2*r*mp^2)/(-L^2*r^2*mp^2+Jp*(Jr+r^2*(mp+mr)));

a2 = (-2*n*Jp*Ke*Kt-2*L*n*r*Ke*Kt*mp)/((-
L^2*r^2*mp^2+Jp*(Jr+r^2*(mp+mr)))*Ra);

a3 = (g*L*Jr*mp*Ra+L*r^2*mp*(g*mp+g*mr)*Ra)/((-
L^2*r^2*mp^2+Jp*(Jr+r^2*(mp+mr)))*Ra);

a4 = (-2*n*Jr*Ke*Kt-2*n*r*Ke*Kt*((L+r)*mp+r*mr))/((-
L^2*r^2*mp^2+Jp*(Jr+r^2*(mp+mr)))*Ra);

b1 = (2*n*Jp*Kt+2*L*n*r*Kt*mp)/((-
L^2*r^2*mp^2+Jp*(Jr+r^2*(mp+mr)))*Ra);

b2 = (2*n*Jr*Kt+2*n*r*Kt*((L+r)*mp+r*mr))/((-
L^2*r^2*mp^2+Jp*(Jr+r^2*(mp+mr)))*Ra);
```

```
%% Modelo Planta

A = [0 0 1;a1 a2 0;a3 a4 0];
B = [0;b1;b2];
C = [1 0 0];
D = [0];
```

```
%% Funcion de transferencia

[num,den] = ss2tf(A,B,C,D)
Gs = tf(num,den)
```

Figura 53. Script de Matlab para calcular la función de transferencia

$$G(s) = \frac{33.86s}{s^3 + 8.172s^2 - 192.1s - 580.5}$$

El comando ss2tf de Matlab es equivalente a realizar las operaciones de la función de transferencia de la página anterior ya que transforma el modelo de estados en numerador y denominador de una función de transferencia.

10 CONTROL PID

El controlador PID es uno de los más utilizados en el control de sistemas realimentados. Algunas de sus ventajas son su sencillez y la capacidad que tiene de dar un buen comportamiento en una gran variedad de situaciones sin la necesidad de conocer con detalle la planta a controlar.

El controlador PID se conoce desde hace bastante tiempo. Se utilizó por primera vez en 1911 y su primer análisis teórico fue en 1922 realizado por Nicolas Minorsky. En aquellos tiempos el control PID era exclusivamente analógico. Sin embargo, resulta sencillo implementar un PID digital en programación y los cálculos que requiere son fáciles y eficientes.

A pesar de la fama que tiene el controlador PID, actualmente no es el mejor controlador disponible. Pero en la mayor parte de los casos es más que suficiente. Por otra parte, la mayoría de los controladores más modernos no dejan de ser versiones mejoradas de un PID como por ejemplo las diversas familias basadas en controladores PID con parámetros adaptativos.

El algoritmo PID está formado por la suma de tres componentes, proporcional, integral y derivativo. Matemáticamente dicho controlador tiene la siguiente formulación:

$$u(t) = K_p e(t) + \frac{K_p}{T_i}\int_0^t e(t)dt + K_p T_d \frac{de(t)}{dt}$$

O la función de transferencia es:

$$\frac{U(s)}{E(s)} = K_p(1 + \frac{1}{T_i s} + T_d s)$$

Cada componente del controlador es independiente de las demás y los tres componentes se suman para obtener la salida del PID. Cada uno cumple una función concreta y mejora cierta parte de la respuesta. Cuando los tres componentes consiguen trabajar juntos, en la proporción adecuada, se obtiene un gran comportamiento.

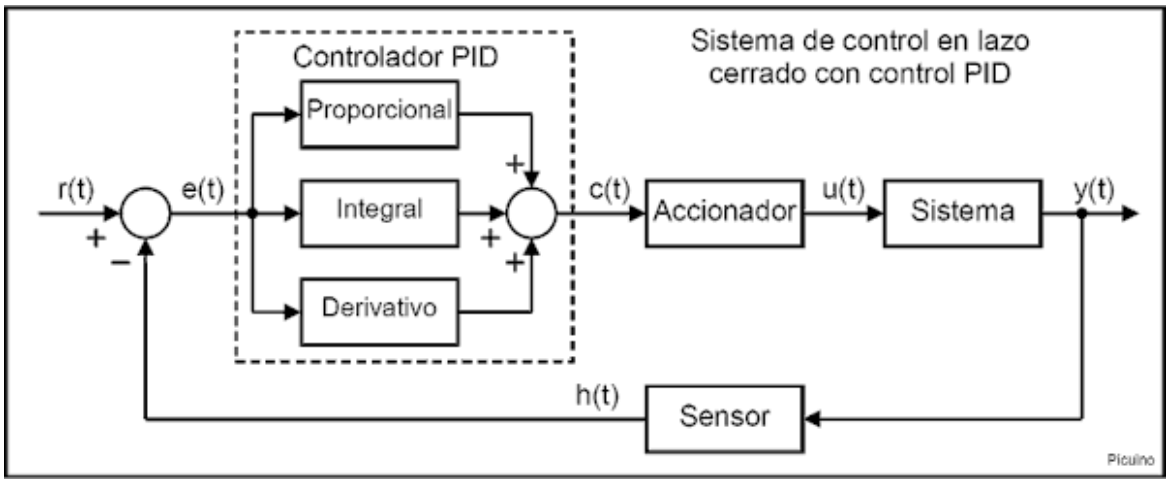


Figura 54. Esquema de control en lazo cerrado con PID

Cada componente tiene un parámetro Kp, Ki y Kd respectivamente. Estos parámetros indican la fuerza que tiene cada componente en el resultado final. Si se ajustan correctamente estos tres parámetros, la respuesta del controlador será buena. No sirve con ajustar cada uno de los parámetros de manera independiente ya que existe una cierta zona dentro de los tres parámetros donde el comportamiento es más o menos decente. Se puede observar de esta manera que la dificultad de un PID está en ajustar los parámetros Kp, Ki y Kd de forma correcta para que el comportamiento final sea bueno.

El componente proporcional se calcula como un factor Kp multiplicado por el error que es la diferencia entre la consigna y el valor real. El factor proporcional tiene una gran influencia en la velocidad de respuesta del sistema. Con una Kp pequeña el sistema va a tardar mucho en alcanzar la consigna. Si se aumenta el valor de Kp, se va a disminuir el tiempo de respuesta. Pero si se aumenta demasiado la Kp se puede sobrecargar la consigna o incluso oscilar y con ello estropear el sistema. Otra característica importante del componente proporcional es que no elimina totalmente el error a largo plazo.

El componente integral aplica una acción que es proporcional a la integral del error a lo largo del tiempo. Responde proporcionalmente a la suma de todos los errores anteriores. Este componente permite al PID eliminar por completo el error a largo plazo. Sin embargo, si el valor de Ki es muy pequeño, el sistema tardará mucho en eliminar el error. Por otro lado, el componente integral tiene tendencia a sobrepasar y oscilar, más incluso que el término proporcional.

El componente derivativo se calcula de manera proporcional a la derivada del error respecto del tiempo en el momento presente. Responde proporcionalmente a la diferencia entre el error actual y el error en el instante anterior. Este componente mejora la respuesta general de muchos sistemas para valores de Kd moderados. Sin embargo, si se aumenta mucho el valor de Kd aparecen comportamientos raros. Además, el componente derivativo responde muy mal al ruido de la medición. El ruido supone variaciones muy rápidas y estas variaciones son amplificadas por este componente. También, a veces este componente pide acciones muy grandes y debido a esto se podría dañar el accionador.

11 FIRMWARE DEL SISTEMA

A continuación, se va a ir explicando detalladamente el programa de Arduino que hace que el robot equilibrista se mantenga estable. El esquema de conexiones del sistema se puede ver en los planos del documento.

```
//Libreria necesaria
#include <Wire.h>

//Direccion I2C de la IMU
#define MPUAddress 0x68

//Conversion de radianes a grados 180/pi
#define RadAGrados 57.295779

//Ratios de conversion del acelerometro y giroscopio
#define AR 16384.0
#define GR 131.0

//La IMU MPU-6050 da los valores en enteros de 16 bits
int16_t AX, AY, AZ, GX, GY, GZ;
```

La primera línea incluye la librería Wire, necesaria para poder realizar la comunicación con la IMU a través del protocolo I2C.

La siguiente línea es la dirección I2C de la IMU, 68 en hexadecimal.

RadAGrados es la conversión de radianes a grados necesaria para calcular el ángulo del acelerómetro. Los ratios de conversión están especificados en la documentación y son unas constantes entre las que se dividen los valores del giroscopio y del acelerómetro para obtener un resultado coherente.

La IMU da los valores en enteros de 16 bits y como Arduino los guarda en menos bits, es necesario declarar las variables que van a almacenar los enteros provenientes de la IMU como un tipo de enteros especiales (int16_t). Dichos valores son los valores en bruto de la IMU.

```
//Angulos necesarios
float Ac;
float Gy;
float Angulo;

//Pines del driver
int IN1 = 9;
int IN2 = 10;
int IN3 = 6;
int IN4 = 5;
int SalI, SalD;
```

En Ac, Gy y Angulo se guardan los ángulos del acelerómetro, giroscopio y el resultado del filtro complementario, respectivamente. Seguidamente, están declarados los pines del driver que van a salidas analógicas PWM para poder controlar la velocidad de los motores de continua. SalI y SalD son dos variables creadas para ser capaces de controlar el sentido de giro de los motores.

```
//Configuracion del PID
unsigned long lastTime;
double Input, Output, Setpoint, output;
double Ierror, lastErr;

double kp = 20;
double ki = 80;
double kd = 0.5;

int SampleTime = 10; //10 ms
double SampleTimeSec = (double)SampleTime / 1000;
double outMin = -255;
double outMax = 255;
```

Se definen las variables necesarias para poder implementar el controlador PID y se introducen los parámetros de ajuste calculados mediante la Sisotool de Matlab. En la variable SampleTime se declara el tiempo de muestreo del

programa que van a ser 10 milisegundos. Dicho tiempo de muestreo es el tiempo que transcurre entre dos mediciones consecutivas, fundamental para obtener datos fiables a través de la IMU. Con las variables outMin y outMax se definen los valores mínimo y máximo de la salida PWM.

```
void Compute()
{
  unsigned long now = millis();
  int timeChange = (now - lastTime);
  //Serial.println(timeChange);
  if (timeChange >= SampleTime)
  {
    //Errores
    double error = Setpoint - Input;
    Ierror += error * SampleTimeSec;
    double dErr = (error - lastErr) / SampleTimeSec;

    //Salida
    Output = kp * error + ki * Ierror + kd * dErr;

    if (Output > outMax)
    {
       Output = outMax;
       //Ierror = 0;
    }
    else if (Output < outMin)
    {
       Output = outMin;
       //Ierror = 0;
    }
    lastErr = error;
    lastTime = now;
  }
```

```
}
```

Función para calcular la salida PWM mediante un controlador PID. Primero se hallan los errores proporcional, integral y derivativo, a partir de estos valores se calcula la salida con el controlador PID y, por último, se aplica el anti wind-up poniendo límites a la salida y se actualizan los valores necesarios. Todo esto ocurre cada 10 ms que es el tiempo de muestreo del sistema.

```
void setup()
{
Wire.begin();
Wire.beginTransmission(MPUAddress);
Wire.write(0x6B);
Wire.write(0);
Wire.endTransmission(true);
Serial.begin(9600);

//Configuracion pines del driver
pinMode (IN1,OUTPUT);
pinMode (IN2,OUTPUT);
pinMode (IN3,OUTPUT);
pinMode (IN4,OUTPUT);

//Configuracion del PID
Setpoint = 0;
Input = Angulo;
}
```

En la función setup primero se inicializa la comunicación por I2C con el sensor IMU y se activa enviando el comando 0. Posteriormente se inicia el puerto serie para observar los resultados.

En la configuración de los pines del driver se establecen como salida ya que van a las salidas PWM de Arduino.

En la configuración del PID se define la referencia que es 0 grados para que sea estable el robot y la entrada que es el ángulo calculado con el filtro complementario.

```
void loop()
{
  //Leer los valores del Acelerometro de la IMU
  Wire.beginTransmission(MPUAddress);
  Wire.write(0x3B); //Pedir el registro 0x3B que corresponde al AX
  Wire.endTransmission(false);
  Wire.requestFrom(MPUAddress,6,true); //A partir del 0x3B, se piden 6 registros

  AX = Wire.read()<<8|Wire.read(); //Cada valor ocupa 2 registros
  AY = Wire.read()<<8|Wire.read();
  AZ = Wire.read()<<8|Wire.read();

  //A partir de los valores del acelerometro, se calcula el angulo
  //con la formula de la tangente
  Ac = atan(-1*(AX/AR)/sqrt(pow((AY/AR),2) + pow((AZ/AR),2)))*RadAGrados;

  //Leer los valores del Giroscopio
  Wire.beginTransmission(MPUAddress);
  Wire.write(0x43); //Pedir el registro 0x43 que corresponde al GX
  Wire.endTransmission(false);
  Wire.requestFrom(MPUAddress,4,true); //A diferencia del Acelerometro, solo se
piden 4 registros

  GX = Wire.read()<<8|Wire.read();  //Cada valor ocupa 2 registros
  GY = Wire.read()<<8|Wire.read();

  //Calculo del angulo del Giroscopio
  Gy = GY / GR;
```

```
  //Se aplica el Filtro Complementario
  Angulo = 0.98 * (Angulo + Gy * 0.010) + 0.02 * Ac;
```

La función loop es la parte más compleja del programa. En ella se leen y se guardan los datos del acelerómetro y del giroscopio de la IMU, se calcula el ángulo, se aplica el filtro complementario y se configura el control del lazo PID como se verá a continuación.

```
//Configuracion lazo PID
  Input = Angulo;
  Compute(); //Funcion que calcula la salida PWM
  if (Output > 0)
  {
    SalI = 5;
    SalD = 9;
    digitalWrite(6,LOW);
    digitalWrite(10,LOW);
  }
  if (Output < 0)
  {
    SalI = 6;
    SalD = 10;
    digitalWrite(5,LOW);
    digitalWrite(9,LOW);
  }
  output = abs(Output);
  analogWrite(SalI,output);
  analogWrite(SalD,output);

  //Mostrar los valores por consola
  Serial.println (Angulo);
  //Serial.print("Salida controlador: "); Serial.println(output);
}
```

La entrada es el ángulo calculado y filtrado, y seguidamente se llama a la función Compute para que calcule la salida Output mediante el controlador PID ya definido anteriormente.

Con el if se controla el sentido de giro de los motores. Si es positiva la salida van a girar hacia un lado y si es negativa en sentido contrario.

En output se guarda el valor absoluto de Output puesto que el microcontrolador no puede enviar ninguna salida PWM de valor negativo. En las dos siguientes líneas se envía a los dos motores la misma salida PWM en valor absoluto.

12 DISEÑO DE CONTROLADOR PID MEDIANTE TÉCNICAS HEURÍSTICAS

Una vez realizado el código de Arduino con el que se pretende estabilizar el robot equilibrista, se procede a diseñar los tres parámetros del controlador PID (Kp, Ki y Kd) para conseguir finalmente la estabilización del sistema.

Este método heurístico consiste en realizar modificaciones sucesivas de las constantes de control hasta conseguir que el sistema real se estabilice en torno a la referencia deseada. En este caso el valor de estabilización es un ángulo de cero grados de inclinación con respecto a la posición vertical.

Se comienza la sintonización con las constantes integral (Ki) y derivativa (Kd) a cero, luego se va dando valores a la constante proporcional (Kp) hasta lograr una oscilación con la menor amplitud posible en torno al punto de ajuste y, a partir de ahí, se regulan los valores de Kd y Ki de forma iterativa hasta lograr la exactitud deseada.

Si es necesario, se varía también Kp, en función de los aportes de la acción integral y derivativa. La mayor dificultad de este método reside en establecer parámetros iniciales, a partir de los cuales se pueda ajustar el controlador.

A continuación, se van a explicar los pasos que se han ido siguiendo para conseguir los parámetros óptimos para la estabilización:

- Primero se dan valores a Kp para conseguir el óptimo, con unas oscilaciones de menor amplitud en torno al ángulo cero deseado. Con valores bajos como 10 el robot no consigue la fuerza necesaria para mantener el equilibrio y se empieza a tambalear, es decir, se queda escasa la acción proporcional. Con valores más altos como 30 las oscilaciones son más fuertes y tampoco se mantiene estable. Con un valor medio como 20 es con el que se consigue una mejor respuesta del sistema.
- Una vez obtenido el valor de Kp, se procede a probar con los valores de Kd y Ki. Primero se prueba con Kp = 20 y Kd = 0.5, sin añadir la acción

integral todavía, y el robot consigue mayor estabilidad que aplicándole solo la acción proporcional.

- Después se añade la acción integral para conseguir eliminar totalmente el error en régimen permanente y que el robot funcione de la mejor manera. Con valores bajos de Ki no consigue estabilizarse por completo hasta que se llega a un valor de 80 con el que el robot funciona perfectamente. Además, se le añaden perturbaciones (pequeños empujones al robot) para comprobar que el diseño es robusto y consigue reaccionar sin caerse al suelo. Por tanto, es un diseño bastante fiable. Los valores de los parámetros del controlador PID son los siguientes: Kp = 20, Kd = 0.5 y Ki = 80.

En la siguiente imagen sacada del Serial Plotter de Arduino se puede ver la variación de ángulo del robot con dichos parámetros. Cuando hay más oscilaciones es que se ha producido alguna pequeña perturbación.

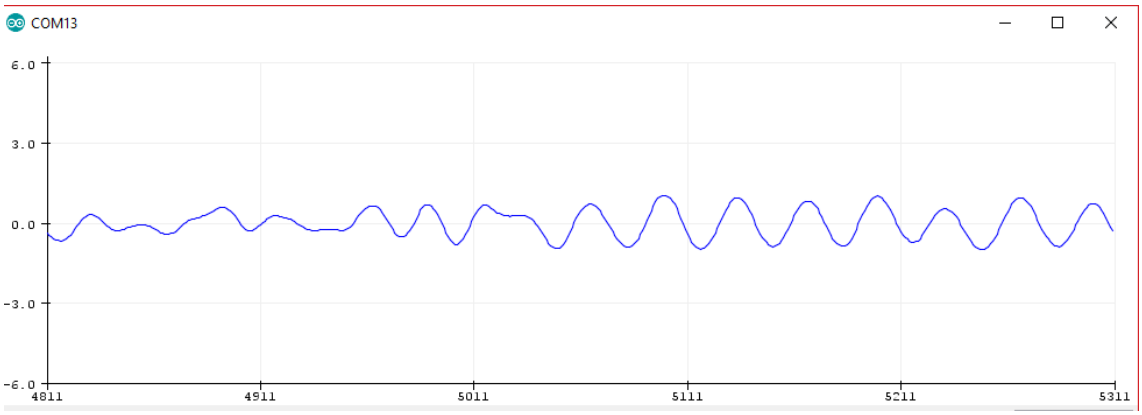


Figura 55. Ángulo de inclinación en Arduino

13 SIMULACIÓN DEL CONTROLADOR PID CON LA HERRAMIENTA SISOTOOL

Una vez diseñado el controlador PID y comprobado que funciona correctamente en el sistema real, se va a introducir junto con la planta del sistema en la herramienta sisotool de Matlab. Esto sirve de comprobación también para ver si el modelo matemático del sistema se ha realizado de manera correcta.

La función de transferencia del sistema ya se ha calculado en el apartado del modelo matemático del sistema. El problema que surge es que esa función de transferencia relaciona radianes con voltios, sin embargo, el controlador PID que se ha diseñado está calculado para una relación de grados con PWM. Por tanto, hay que transformar la función de transferencia del sistema a esas unidades para poder simular correctamente el funcionamiento del controlador.

$$Ps \rightarrow Gs[\frac{rad}{V}] * \frac{9V}{255PWM} * \frac{180grados}{\pi rad}$$

Multiplicando la función de transferencia por esos factores de conversión obtenemos la planta deseada para introducir en la sisotool. En la siguiente imagen se puede ver el código necesario para introducir en Matlab.

```
Ps = zpk(Gs)*9/255*180/pi
Kd = 0.5
Kp = 20
Ki = 80
Cs = (Kd*s^2+Kp*s+Ki)/s
sisotool(Ps,Cs)
```

Figura 56. Código de Matlab para abrir la sisotool

Cuando se abre la sisotool se observan las siguientes gráficas. A la izquierda tenemos la respuesta al escalón, arriba a la derecha el lugar de las raíces en lazo cerrado y abajo a la derecha la respuesta al impulso del sistema.

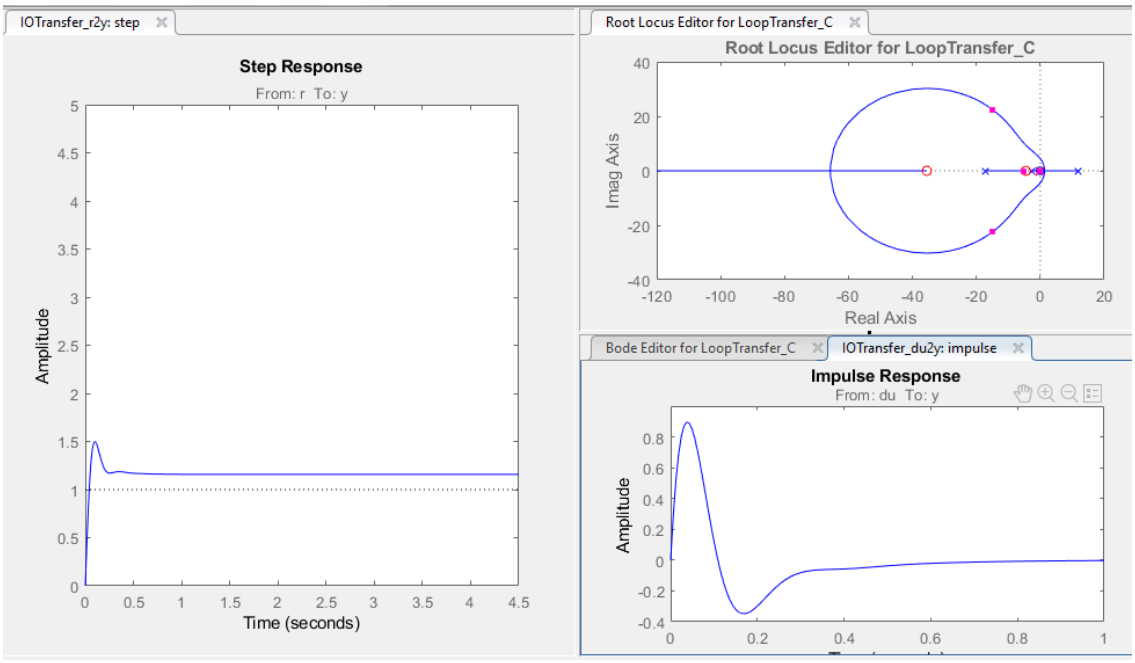


Figura 57. Gráficas de la sisotool

Con la respuesta al escalón y la respuesta al impulso podemos ver que el sistema es estable en régimen permanente. Tarda en estabilizarse unos 0.3 segundos. Por tanto, la función de transferencia obtenida a partir del modelo matemático está bien calculada ya que con el mismo controlador PID que se ha empleado para estabilizar el sistema real se consigue estabilizar también la simulación del sistema.

14 CONCLUSIONES Y LÍNEAS FUTURAS

La realización de este proyecto ha sido una forma muy interesante y amena de poder realizar el control de un sistema inestable y no lineal a través de modelado y simulación, y posteriormente tener la opción de trasladar ese estudio a un robot real, pudiendo observar su comportamiento en tiempo real.

También, ha resultado muy satisfactorio y útil el tener que elaborar desde cero todo el diseño mecánico y electrónico del robot equilibrista. Además, ha habido que repasar y estudiar algunas leyes físicas estudiadas en asignaturas de primer año ya que eran necesarias para poder realizar correctamente el modelo matemático del robot con el que posteriormente se diseña el control de estabilidad.

Cabe destacar también la oportunidad de poder conocer dispositivos electrónicos que anteriormente se desconocían como por ejemplo el sensor IMU MPU-6050. Dicho sensor tiene un funcionamiento bastante complejo y después de investigar en profundidad sobre él se ha aprendido a leer datos provenientes del giroscopio y del acelerómetro en grados de inclinación a través de diversos cálculos trigonométricos y empleando el filtro complementario para obtener lecturas más óptimas y fiables.

Por otra parte, también se han adquirido nuevos conocimientos sobre el microcontrolador Arduino, empleado en el proyecto. Es una plataforma diseñada para facilitar el uso de la electrónica en proyectos multidisciplinares a nivel europeo y sobre todo destaca su programación en código abierto perfecta para poder iniciarse en el mundo de la electrónica de una manera sencilla y entretenida.

La elaboración de este proyecto requiere también tener que repasar la teoría vista en varias asignaturas del grado como por ejemplo Ingeniería de Control, Informática Industrial, Diseño de Aplicaciones Electrónicas, Mecánica, Electrónica de Potencia, Electrónica Digital…

Para finalizar se han pensado además una serie de futuras mejoras. Algunas de ellas simplemente mejorando algunas prestaciones ya presentes y otras aumentando las funcionalidades del robot equilibrista.

Las posibles mejoras se citan a continuación:

- Sustituir las seis pilas de 1,5V por baterías LiPo. Estas son unas baterías de polímetro de iones de litio, que a pesar del peligro que tienen (pueden arder si no se tratan de una forma adecuada), son indicadas para este tipo de aplicaciones. Una sola batería LiPo es capaz de suministrar la energía necesaria para la demanda de consumo de los motores cosa que a veces las pilas puede que no tengan suficiente potencia para mover los motores.
- Reemplazar los motores de corriente continua por otro tipo de motores como por ejemplo brushless o paso a paso que permitan un control mucho más preciso de la velocidad de los mismos.
- Utilizar el algoritmo del filtro de Kalman en la obtención del ángulo de inclinación. De esta manera se ganaría algo de rapidez a la hora de obtener el ángulo con respecto al filtro complementario, y así se tiene más tiempo para la regulación y el control.
- Diseñar y construir la estructura del robot equilibrista con otro tipo de material, por ejemplo, con metacrilato que es un material resistente y no muy pesado.
- Emplear una conexión inalámbrica para dirigir y manejar el robot desde un smartphone, y que se pueda mover siguiendo el recorrido deseado por el usuario.

15 BIBLIOGRAFÍA

15.1 LIBROS

Cannon, R., Dynamics of Physical Systems. New York: McGraw-Hill Book Company, 1967.

Katsuhiko Ogata. “Ingeniería de control moderna”. Pearson Educación, S.A., Madrid, (2010): 722- 786. ISBN: 978-84-8322-660-5.

MathWorks, Inc., The Student Edition of MATLAB, version 5. Upper Saddle River, NJ: Prentice Hall, 1997.

Ogata, K., Discrete-Time Control Systems, 2nd ed. Upper Saddle River, NJ: Prentice Hall, 1995.

S. Crandall, D. Karnopp, E. Kurtz, and D. Pridmore-Brown. Dynamics of Mechanical and Electromechanical Systems. Krieger Publishing Company, Malabar, Florida, 1968.

Ogata, K., Designing Linear Control Systems with MATLAB. Upper Saddle River, NJ: Prentice Hall, 1994.

Luenberger, D. G., «An Introduction to Observers», IEEE Trans. Automatic Control, AC-16 (1971), pp.596-602.

Cunningham, W. J., Introduction to Nonlinear Analysis. New York: McGraw-Hill Book Company, 1958.

Feng, Q., Yamafuji, K., “Design and simulation of control systems of an inverted pendulum”, Robotica, 6 (3), pp. 235-241 (1988).

Grasser, F., D'Arrigo, A., Colombi, S., Rufer, A.C., “JOE: A mobile, inverted pendulum”, IEEE Transactions on Industrial Electronics, 49 (1), pp. 107-114 (2002)

Brogan, W. L., Modern Control Theory. Upper Saddle River, NJ: Prentice Hall, 1985.

Evans, W. R., «Control System Synthesis by Root Locus Method», AIEE Trans Part II, 69 (1950), pp. 6-9.

Dorf, Richard C.; Bishop, Robert H. Sistemas de control moderno. 10ª ed. Madrid [etc.]: Prentice Hall, cop. 2005. ISBN 8420544019.

Kuo, Benjamin C. Sistemas automáticos de control. 9ª ed. México: Compañía Editorial Continental, 1991. ISBN 9682611393.

15.2 PÁGINAS DE INTERNET

Guía IMU MPU-6050 (Acelerómetro y Giroscopio), (2014-10-15), url: http://www.starlino.com/imu_guide.html

Arduino web site, version dated (2014-07-15), url: http://www.arduino.cc/

Teoría Puente en H. (2014-08-10), url: http://www.taringa.net/post/cienciaeducacion/8662220/Que-es-y-como-funciona-un-puente-H.html

Tutorial: Uso de Driver L298N para motores DC y paso a paso con Arduino. (2014-07-14), url: http://electronilab.co/tutoriales/tutorial-de-uso-driver-dual-l298n-para-motores-dc-y-paso-a-pasocon-arduino/

Wikipedia. Modulación por ancho de pulso, versión (2014-09-03), url: http://en.wikipedia.org/wiki/pulse-width-modulation, 2012.

Wikipedia. Modelado de un motor CC. url: http://isa.uniovi.es/ISAwiki/index.php/Modelado_de_un_motor_CC

Teoría de control en Arduino: El controlador PID. url: https://www.luisllamas.es/teoria-de-control-en-arduino-el-controlador-pid/

Modelado de la función de transferencia a partir del espacio de estados. url: http://blog.utp.edu.co/docenciaedwin/files/2011/05/Funci%C3%B3n-de-transferencia-a-partir-del-espacio-de-estados.pdf

Foros de Electrónica. Diseño de PID para planta de tercer orden inestable. url: https://www.forosdeelectronica.com/threads/dise%C3%B1o-pid-para-planta-de-tercer-orden-inestable.56425/

15.3 TRABAJOS FIN DE GRADO PUBLICADOS

Diseño y control de un sistema electromecánico inestable de dos grados de libertad: aplicación al caso de un péndulo invertido. Universidad Politécnica de Cataluña. Autor: Eugenio Molis Merino.

Desarrollo de un robot autoequilibrado basado en Arduino con motores paso a paso. Universidad de Sevilla. Autor: José Antonio Borja Conde.

Diseño, construcción y control de un robot balancín. Universidad Nacional de Cuyo. Autor: Santiago Javier Pincin

Diseño e implementación en PCB de un robot auto-balanceado mediante Arduino con módulo inalámbrico. Universidad Pública de Navarra. Autor: Víctor Esteban Falconi Loja.

Diseño y construcción de un robot auto-balanceado mediante Arduino. Universidad Pública de Navarra. Autor: Ander Gracia Moisés.

Diseño e implementación de robot segway. Universidad Pública de Navarra. Autor: Ion Irigoyen Martínez.

ANEXOS

TRABAJO FIN DE GRADO

UNIVERSIDAD DE LA RIOJA

GRADO EN INGENIERÍA ELECTRÓNICA INDUSTRIAL Y AUTOMÁTICA

CURSO 2018/19

REALIZADO POR: JORGE BUENO GIL

TUTORES DEL PROYECTO: JAVIER RICO AZAGRA
MONTSERRAT GIL MARTÍNEZ

ÍNDICE ANEXOS

16 ANEXOS

16.1 CARACTERÍSTICAS TÉCNICAS DE LA IMU MPU-6050

InvenSense	MPU-6000/MPU-6050 Product Specification	Document Number: PS-MPU-6000A-00 Revision: 3.4 Release Date: 08/19/2013

6 Electrical Characteristics

6.1 Gyroscope Specifications

VDD = 2.375V-3.46V, VLOGIC (MPU-6050 only) = 1.8V±5% or VDD, T_A = 25°C

PARAMETER	CONDITIONS	MIN	TYP	MAX	UNITS	NOTES
GYROSCOPE SENSITIVITY						
Full-Scale Range	FS_SEL=0		±250		°/s	
	FS_SEL=1		±500		°/s	
	FS_SEL=2		±1000		°/s	
	FS_SEL=3		±2000		°/s	
Gyroscope ADC Word Length			16		bits	
Sensitivity Scale Factor	FS_SEL=0		131		LSB/(°/s)	
	FS_SEL=1		65.5		LSB/(°/s)	
	FS_SEL=2		32.8		LSB/(°/s)	
	FS_SEL=3		16.4		LSB/(°/s)	
Sensitivity Scale Factor Tolerance	25°C	-3		+3	%	
Sensitivity Scale Factor Variation Over Temperature			±2		%	
Nonlinearity	Best fit straight line; 25°C		0.2		%	
Cross-Axis Sensitivity			±2		%	
GYROSCOPE ZERO-RATE OUTPUT (ZRO)						
Initial ZRO Tolerance	25°C		±20		°/s	
ZRO Variation Over Temperature	-40°C to +85°C		±20		°/s	
Power-Supply Sensitivity (1-10Hz)	Sine wave, 100mVpp; VDD=2.5V		0.2		°/s	
Power-Supply Sensitivity (10 - 250Hz)	Sine wave, 100mVpp; VDD=2.5V		0.2		°/s	
Power-Supply Sensitivity (250Hz - 100kHz)	Sine wave, 100mVpp; VDD=2.5V		4		°/s	
Linear Acceleration Sensitivity	Static		0.1		°/s/g	
SELF-TEST RESPONSE						
Relative	Change from factory trim	-14		14	%	1
GYROSCOPE NOISE PERFORMANCE	**FS_SEL=0**					
Total RMS Noise	DLPFCFG=2 (100Hz)		0.05		°/s-rms	
Low-frequency RMS noise	Bandwidth 1Hz to10Hz		0.033		°/s-rms	
Rate Noise Spectral Density	At 10Hz		0.005		°/s/ √ Hz	
GYROSCOPE MECHANICAL FREQUENCIES						
X-Axis		30	33	36	kHz	
Y-Axis		27	30	33	kHz	
Z-Axis		24	27	30	kHz	
LOW PASS FILTER RESPONSE						
	Programmable Range	5		256	Hz	
OUTPUT DATA RATE						
	Programmable	4		8,000	Hz	
GYROSCOPE START-UP TIME	**DLPFCFG=0**					
ZRO Settling (from power-on)	to ±1°/s of Final		30		ms	

1. Please refer to the following document for further information on Self-Test: *MPU-6000/MPU-6050 Register Map and Descriptions*

6.2 Accelerometer Specifications

VDD = 2.375V-3.46V, VLOGIC (MPU-6050 only) = 1.8V±5% or VDD, T_A = 25°C

PARAMETER	CONDITIONS	MIN	TYP	MAX	UNITS	NOTES
ACCELEROMETER SENSITIVITY						
Full-Scale Range	AFS_SEL=0		±2		*g*	
	AFS_SEL=1		±4		*g*	
	AFS_SEL=2		±8		*g*	
	AFS_SEL=3		±16		*g*	
ADC Word Length	Output in two's complement format		16		bits	
Sensitivity Scale Factor	AFS_SEL=0		16,384		LSB/*g*	
	AFS_SEL=1		8,192		LSB/*g*	
	AFS_SEL=2		4,096		LSB/*g*	
	AFS_SEL=3		2,048		LSB/*g*	
Initial Calibration Tolerance			±3		%	
Sensitivity Change vs. Temperature	AFS_SEL=0, -40°C to +85°C		±0.02		%/°C	
Nonlinearity	Best Fit Straight Line		0.5		%	
Cross-Axis Sensitivity			±2		%	
ZERO-G OUTPUT						
Initial Calibration Tolerance	X and Y axes		±50		mg	1
	Z axis		±80		mg	
Zero-G Level Change vs. Temperature	X and Y axes, 0°C to +70°C		±35			
	Z axis, 0°C to +70°C		±60		mg	
SELF TEST RESPONSE						
Relative	Change from factory trim	-14		14	%	2
NOISE PERFORMANCE						
Power Spectral Density	@10Hz, AFS_SEL=0 & ODR=1kHz		400		µg/√Hz	
LOW PASS FILTER RESPONSE						
	Programmable Range	5		260	Hz	
OUTPUT DATA RATE						
	Programmable Range	4		1,000	Hz	
INTELLIGENCE FUNCTION INCREMENT			32		mg/LSB	

1. Typical zero-g initial calibration tolerance value after MSL3 preconditioning
2. Please refer to the following document for further information on Self-Test: *MPU-6000/MPU-6050 Register Map and Descriptions*

6.3 Electrical and Other Common Specifications

VDD = 2.375V-3.46V, VLOGIC (MPU-6050 only) = 1.8V±5% or VDD, T_A = 25°C

PARAMETER	CONDITIONS	MIN	TYP	MAX	Units	Notes
TEMPERATURE SENSOR						
Range			-40 to +85		°C	
Sensitivity	Untrimmed		340		LSB/°C	
Temperature Offset	35°C		-521		LSB	
Linearity	Best fit straight line (-40°C to +85°C)		±1		°C	
VDD POWER SUPPLY						
Operating Voltages		2.375		3.46	V	
Normal Operating Current	Gyroscope + Accelerometer + DMP		3.9		mA	
	Gyroscope + Accelerometer (DMP disabled)		3.8		mA	
	Gyroscope + DMP (Accelerometer disabled)		3.7		mA	
	Gyroscope only (DMP & Accelerometer disabled)		3.6		mA	
	Accelerometer only (DMP & Gyroscope disabled)		500		µA	
Accelerometer Low Power Mode Current	1.25 Hz update rate		10		µA	
	5 Hz update rate		20		µA	
	20 Hz update rate		70		µA	
	40 Hz update rate		140		µA	
Full-Chip Idle Mode Supply Current			5		µA	
Power Supply Ramp Rate	Monotonic ramp. Ramp rate is 10% to 90% of the final value			100	ms	
VLOGIC REFERENCE VOLTAGE	MPU-6050 only					
Voltage Range	VLOGIC must be ≤VDD at all times	1.71		VDD	V	
Power Supply Ramp Rate	Monotonic ramp. Ramp rate is 10% to 90% of the final value			3	ms	
Normal Operating Current			100		µA	
TEMPERATURE RANGE						
Specified Temperature Range	Performance parameters are not applicable beyond Specified Temperature Range	-40		+85	°C	

6.4 Electrical Specifications, Continued

VDD = 2.375V-3.46V, VLOGIC (MPU-6050 only) = 1.8V±5% or VDD, T_A = 25°C

PARAMETER	CONDITIONS	MIN	TYP	MAX	Units	Notes
SERIAL INTERFACE						
SPI Operating Frequency, All Registers Read/Write	MPU-6000 only, Low Speed Characterization		100 ±10%		kHz	
	MPU-6000 only, High Speed Characterization		1 ±10%		MHz	
SPI Operating Frequency, Sensor and Interrupt Registers Read Only	MPU-6000 only		20 ±10%		MHz	
I^2C Operating Frequency	All registers, Fast-mode			400	kHz	
	All registers, Standard-mode			100	kHz	
I^2C ADDRESS	AD0 = 0		1101000			
	AD0 = 1		1101001			
DIGITAL INPUTS (SDI/SDA, AD0, SCLK/SCL, FSYNC, /CS, CLKIN)						
V_{IH}, High Level Input Voltage	MPU-6000	0.7*VDD			V	
	MPU-6050	0.7*VLOGIC			V	
V_{IL}, Low Level Input Voltage	MPU-6000			0.3*VDD	V	
	MPU-6050			0.3*VLOGIC	V	
C_I, Input Capacitance			< 5		pF	
DIGITAL OUTPUT (SDO, INT)						
V_{OH}, High Level Output Voltage	R_{LOAD}=1MΩ; MPU-6000	0.9*VDD			V	
	R_{LOAD}=1MΩ; MPU-6050	0.9*VLOGIC			V	
V_{OL1}, LOW-Level Output Voltage	R_{LOAD}=1MΩ; MPU-6000			0.1*VDD	V	
	R_{LOAD}=1MΩ; MPU-6050			0.1*VLOGIC	V	
$V_{OL.INT1}$, INT Low-Level Output Voltage	OPEN=1, 0.3mA sink Current			0.1	V	
Output Leakage Current	OPEN=1		100		nA	
t_{INT}, INT Pulse Width	LATCH_INT_EN=0		50		µs	

6.5 Electrical Specifications, Continued

Typical Operating Circuit of Section 7.2, VDD = 2.375V-3.46V, VLOGIC (MPU-6050 only) = 1.8V±5% or VDD, T_A = 25°C

Parameters	Conditions	Typical	Units	Notes
Primary I^2C I/O (SCL, SDA)				
V_{IL}, LOW-Level Input Voltage	MPU-6000	-0.5 to 0.3*VDD	V	
V_{IH}, HIGH-Level Input Voltage	MPU-6000	0.7*VDD to VDD + 0.5V	V	
V_{hys}, Hysteresis	MPU-6000	0.1*VDD	V	
VIL, LOW Level Input Voltage	MPU-6050	-0.5V to 0.3*VLOGIC	V	
VIH, HIGH-Level Input Voltage	MPU-6050	0.7*VLOGIC to VLOGIC + 0.5V	V	
Vhys, Hysteresis	MPU-6050	0.1*VLOGIC	V	
V_{OL1}, LOW-Level Output Voltage	3mA sink current	0 to 0.4	V	
I_{OL}, LOW-Level Output Current	V_{OL} = 0.4V	3	mA	
	V_{OL} = 0.6V	5	mA	
Output Leakage Current		100	nA	
t_{of}, Output Fall Time from V_{IHmax} to V_{ILmax}	C_b bus capacitance in pF	20+0.1C_b to 250	ns	
C_i, Capacitance for Each I/O pin		< 10	pF	
Auxiliary I^2C I/O (AUX_CL, AUX_DA)	**MPU-6050: *AUX_VDDIO*=0**			
V_{IL}, LOW-Level Input Voltage		-0.5V to 0.3*VLOGIC	V	
V_{IH}, HIGH-Level Input Voltage		0.7*VLOGIC to VLOGIC + 0.5V	V	
V_{hys}, Hysteresis		0.1*VLOGIC	V	
V_{OL1}, LOW-Level Output Voltage	VLOGIC > 2V; 1mA sink current	0 to 0.4	V	
V_{OL3}, LOW-Level Output Voltage	VLOGIC < 2V; 1mA sink current	0 to 0.2*VLOGIC	V	
I_{OL}, LOW-Level Output Current	V_{OL} = 0.4V	1	mA	
	V_{OL} = 0.6V	1	mA	
Output Leakage Current		100	nA	
t_{of}, Output Fall Time from V_{IHmax} to V_{ILmax}	C_b bus capacitance in pF	20+0.1C_b to 250	ns	
C_i, Capacitance for Each I/O pin		< 10	pF	

6.6 Electrical Specifications, Continued

Typical Operating Circuit of Section 7.2, VDD = 2.375V-3.46V, VLOGIC (MPU-6050 only) = 1.8V±5% or VDD, T_A = 25°C

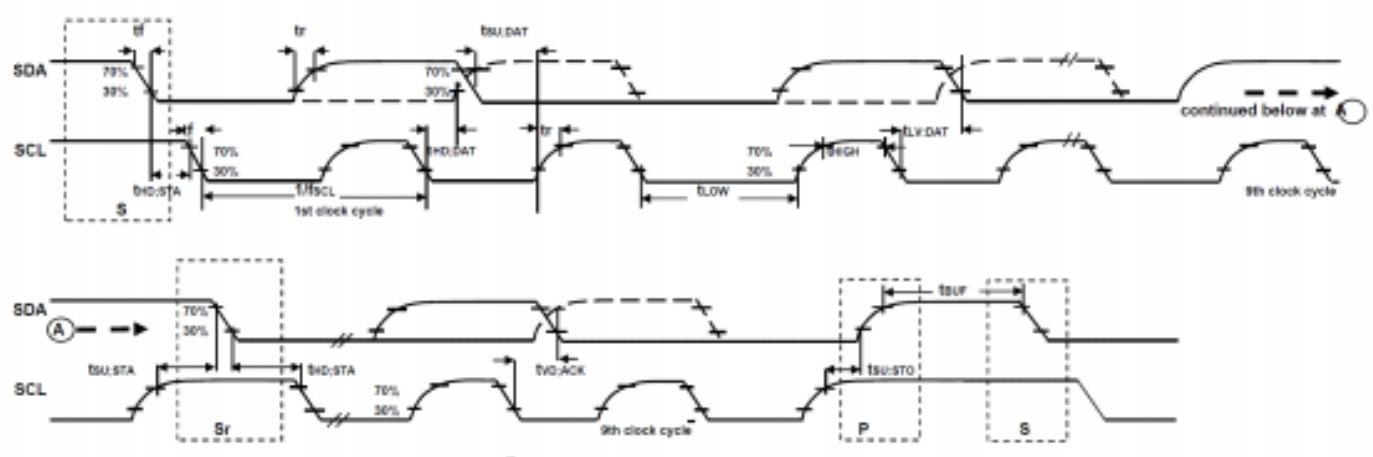
Parameters	Conditions	Min	Typical	Max	Units	Notes
INTERNAL CLOCK SOURCE	**CLK_SEL=0,1,2,3**					
Gyroscope Sample Rate, Fast	DLPFCFG=0 SAMPLERATEDIV = 0		8		kHz	
Gyroscope Sample Rate, Slow	DLPFCFG=1,2,3,4,5, or 6 SAMPLERATEDIV = 0		1		kHz	
Accelerometer Sample Rate			1		kHz	
Clock Frequency Initial Tolerance	CLK_SEL=0, 25°C	-5		+5	%	
	CLK_SEL=1,2,3; 25°C	-1		+1	%	
Frequency Variation over Temperature	CLK_SEL=0		-15 to +10		%	
	CLK_SEL=1,2,3		±1		%	
PLL Settling Time	CLK_SEL=1,2,3		1	10	ms	
EXTERNAL 32.768kHz CLOCK	**CLK_SEL=4**					
External Clock Frequency			32.768		kHz	
External Clock Allowable Jitter	Cycle-to-cycle rms		1 to 2		µs	
Gyroscope Sample Rate, Fast	DLPFCFG=0 SAMPLERATEDIV = 0		8.192		kHz	
Gyroscope Sample Rate, Slow	DLPFCFG=1,2,3,4,5, or 6 SAMPLERATEDIV = 0		1.024		kHz	
Accelerometer Sample Rate			1.024		kHz	
PLL Settling Time			1	10	ms	
EXTERNAL 19.2MHz CLOCK	**CLK_SEL=5**					
External Clock Frequency			19.2		MHz	
Gyroscope Sample Rate	Full programmable range	3.9		8000	Hz	
Gyroscope Sample Rate, Fast Mode	DLPFCFG=0 SAMPLERATEDIV = 0		8		kHz	
Gyroscope Sample Rate, Slow Mode	DLPFCFG=1,2,3,4,5, or 6 SAMPLERATEDIV = 0		1		kHz	
Accelerometer Sample Rate			1		kHz	
PLL Settling Time			1	10	ms	

6.7 I^2C Timing Characterization

Typical Operating Circuit of Section 7.2, VDD = 2.375V-3.46V, VLOGIC (MPU-6050 only) = 1.8V±5% or VDD, T_A = 25°C

Parameters	Conditions	Min	Typical	Max	Units	Notes
I^2C TIMING	**I^2C FAST-MODE**					
f_{SCL}, SCL Clock Frequency				400	kHz	
$t_{HD.STA}$, (Repeated) START Condition Hold Time		0.6			µs	
t_{LOW}, SCL Low Period		1.3			µs	
t_{HIGH}, SCL High Period		0.6			µs	
$t_{SU.STA}$, Repeated START Condition Setup Time		0.6			µs	
$t_{HD.DAT}$, SDA Data Hold Time		0			µs	
$t_{SU.DAT}$, SDA Data Setup Time		100			ns	
t_r, SDA and SCL Rise Time	C_b bus cap. from 10 to 400pF	20+0.1C_b		300	ns	
t_f, SDA and SCL Fall Time	C_b bus cap. from 10 to 400pF	20+0.1C_b		300	ns	
$t_{SU.STO}$, STOP Condition Setup Time		0.6			µs	
t_{BUF}, Bus Free Time Between STOP and START Condition		1.3			µs	
C_b, Capacitive Load for each Bus Line			< 400		pF	
$t_{VD.DAT}$, Data Valid Time				0.9	µs	
$t_{VD.ACK}$, Data Valid Acknowledge Time				0.9	µs	

Note: Timing Characteristics apply to both Primary and Auxiliary I^2C Bus



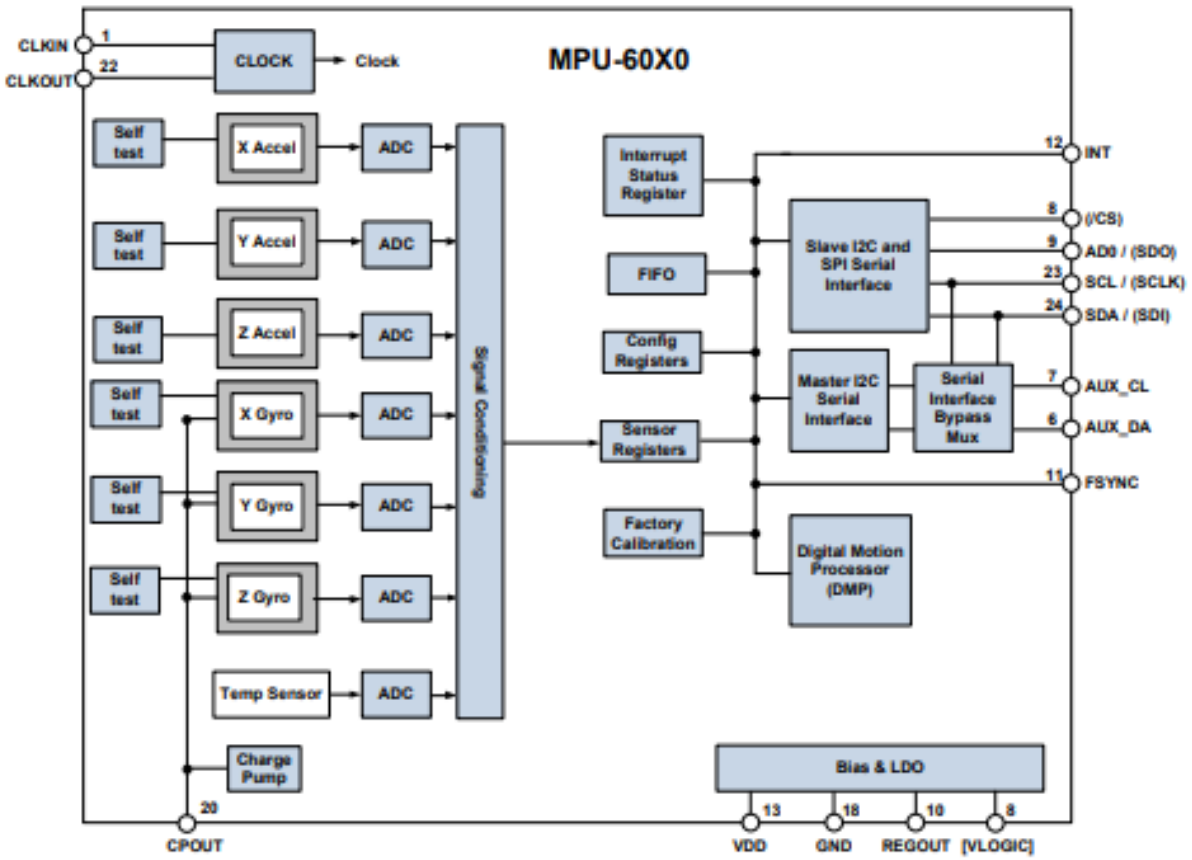
I^2C Bus Timing Diagram

6.9 Absolute Maximum Ratings

Stress above those listed as "Absolute Maximum Ratings" may cause permanent damage to the device. These are stress ratings only and functional operation of the device at these conditions is not implied. Exposure to the absolute maximum ratings conditions for extended periods may affect device reliability.

Parameter	Rating
Supply Voltage, VDD	-0.5V to +6V
VLOGIC Input Voltage Level (MPU-6050)	-0.5V to VDD + 0.5V
REGOUT	-0.5V to 2V
Input Voltage Level (CLKIN, AUX_DA, AD0, FSYNC, INT, SCL, SDA)	-0.5V to VDD + 0.5V
CPOUT (2.5V ≤ VDD ≤ 3.6V)	-0.5V to 30V
Acceleration (Any Axis, unpowered)	10,000g for 0.2ms
Operating Temperature Range	-40°C to +105°C
Storage Temperature Range	-40°C to +125°C
Electrostatic Discharge (ESD) Protection	2kV (HBM); 250V (MM)
Latch-up	JEDEC Class II (2),125°C ±100mA

7.5 Block Diagram

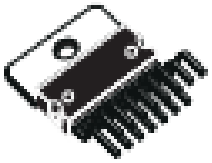


Note: Pin names in round brackets ( ) apply only to MPU-6000
Pin names in square brackets [ ] apply only to MPU-6050

16.2 CARACTERÍSTICAS TÉCNICAS DEL DRIVER L298N

DUAL FULL-BRIDGE DRIVER

- OPERATING SUPPLY VOLTAGE UP TO 46 V
- TOTAL DC CURRENT UP TO 4 A
- LOW SATURATION VOLTAGE
- OVERTEMPERATURE PROTECTION
- LOGICAL "0" INPUT VOLTAGE UP TO 1.5 V (HIGH NOISE IMMUNITY)



Multiwatt15

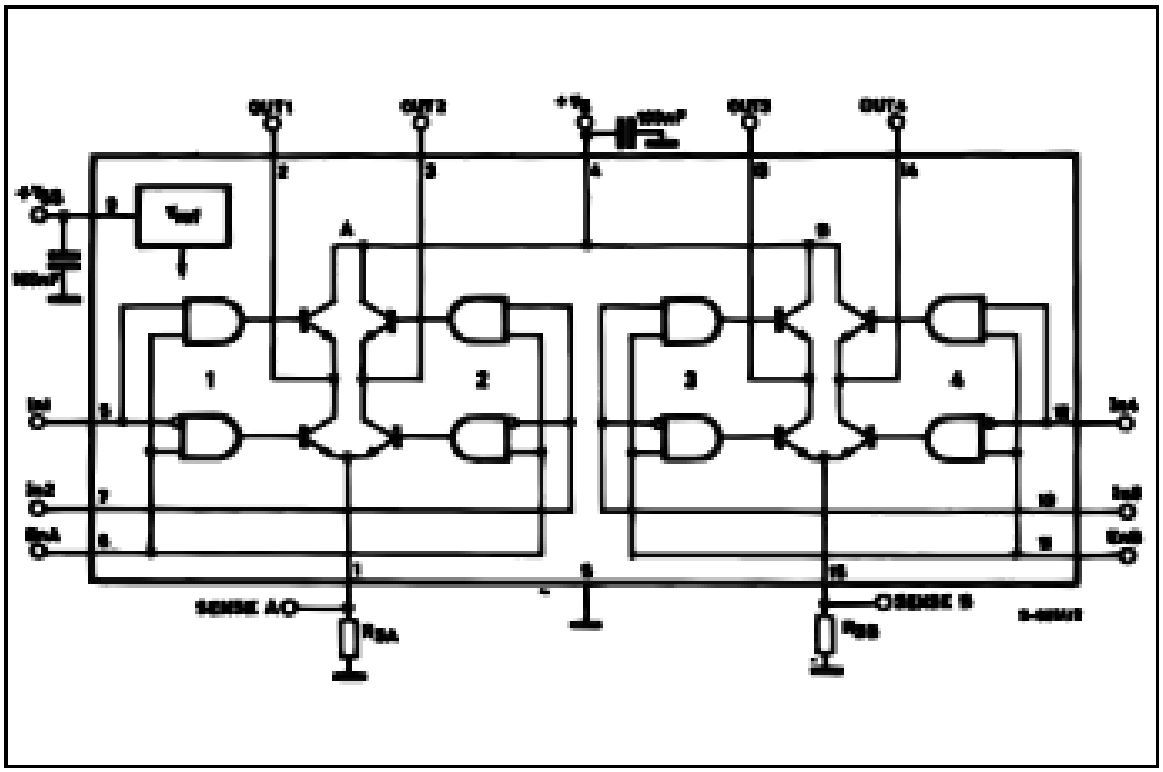
PowerSO20

ORDERING NUMBERS : L298N (Multiwatt Vert.)
L298HN (Multiwatt Horiz.)
L298P (PowerSO20)

DESCRIPTION

The L298 is an integrated monolithic circuit in a 15-lead Multiwatt and PowerSO20 packages. It is a high voltage, high current dual full-bridge driver designed to accept standard TTL logic levels and drive inductive loads such as relays, solenoids, DC and stepping motors. Two enable inputs are provided to enable or disable the device independently of the input signals. The emitters of the lower transistors of each bridge are connected together and the corresponding external terminal can be used for the connection of an external sensing resistor. An additional supply input is provided so that the logic works at a lower voltage.

BLOCK DIAGRAM

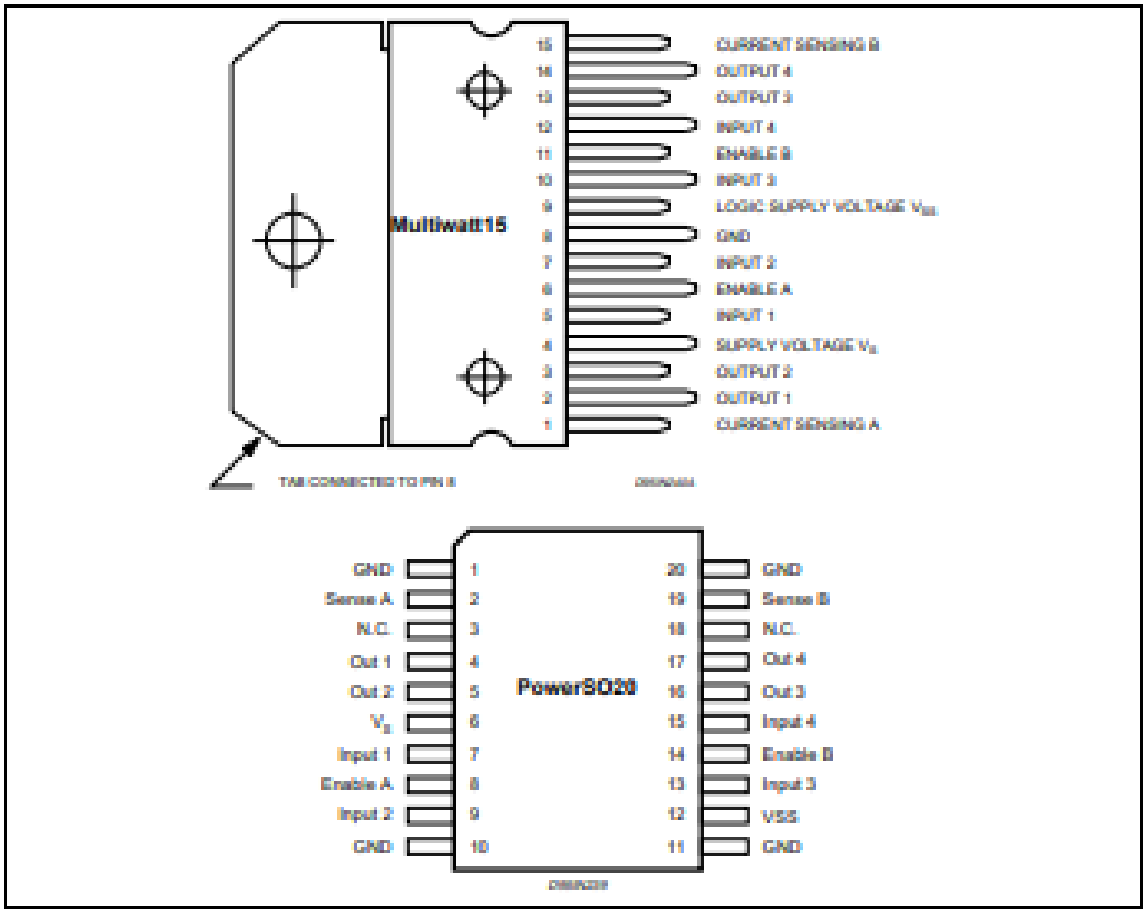


L298

ABSOLUTE MAXIMUM RATINGS

Symbol	Parameter	Value	Unit
V_S	Power Supply	50	V
V_{SS}	Logic Supply Voltage	7	V
V_i,V_{en}	Input and Enable Voltage	–0.3 to 7	V
I_O	Peak Output Current (each Channel) – Non Repetitive (t = 100µs) –Repetitive (80% on –20% off; t_{on} = 10ms) –DC Operation	 3 2.5 2	 A A A
V_{sens}	Sensing Voltage	–1 to 2.3	V
P_{tot}	Total Power Dissipation (T_{case} = 75°C)	25	W
T_{op}	Junction Operating Temperature	–25 to 130	°C
T_{stg}, T_j	Storage and Junction Temperature	–40 to 150	°C

PIN CONNECTIONS (top view)



THERMAL DATA

Symbol	Parameter		PowerSO20	Multiwatt15	Unit
$R_{th\ j\text{-}case}$	Thermal Resistance Junction-case	Max.	–	3	°C/W
$R_{th\ j\text{-}amb}$	Thermal Resistance Junction-ambient	Max.	13 (*)	35	°C/W

(*) Mounted on aluminum substrate

PIN FUNCTIONS (refer to the block diagram)

MW.15	PowerSO	Name	Function
1;15	2;19	Sense A; Sense B	Between this pin and ground is connected the sense resistor to control the current of the load.
2;3	4;5	Out 1; Out 2	Outputs of the Bridge A; the current that flows through the load connected between these two pins is monitored at pin 1.
4	6	V_S	Supply Voltage for the Power Output Stages. A non-inductive 100nF capacitor must be connected between this pin and ground.
5;7	7;9	Input 1; Input 2	TTL Compatible Inputs of the Bridge A.
6;11	8;14	Enable A; Enable B	TTL Compatible Enable Input: the L state disables the bridge A (enable A) and/or the bridge B (enable B).
8	1,10,11,20	GND	Ground.
9	12	VSS	Supply Voltage for the Logic Blocks. A100nF capacitor must be connected between this pin and ground.
10; 12	13;15	Input 3; Input 4	TTL Compatible Inputs of the Bridge B.
13; 14	16;17	Out 3; Out 4	Outputs of the Bridge B. The current that flows through the load connected between these two pins is monitored at pin 15.
–	3;18	N.C.	Not Connected

ELECTRICAL CHARACTERISTICS (V_S = 42V; V_{SS} = 5V, T_j = 25°C; unless otherwise specified)

Symbol	Parameter	Test Conditions		Min.	Typ.	Max.	Unit
V_S	Supply Voltage (pin 4)	Operative Condition		V_{IH} +2.5		46	V
V_{SS}	Logic Supply Voltage (pin 9)			4.5	5	7	V
I_S	Quiescent Supply Current (pin 4)	V_{en} = H; I_L = 0	V_i = L V_i = H		13 50	22 70	mA mA
		V_{en} = L	V_i = X			4	mA
I_{SS}	Quiescent Current from V_{SS} (pin 9)	V_{en} = H; I_L = 0	V_i = L V_i = H		24 7	36 12	mA mA
		V_{en} = L	V_i = X			6	mA
V_{il}	Input Low Voltage (pins 5, 7, 10, 12)			–0.3		1.5	V
V_{ih}	Input High Voltage (pins 5, 7, 10, 12)			2.3		VSS	V
I_{il}	Low Voltage Input Current (pins 5, 7, 10, 12)	V_i = L				–10	µA
I_{ih}	High Voltage Input Current (pins 5, 7, 10, 12)	Vi = H ≤ V_{SS} –0.6V			30	100	µA
V_{en} = L	Enable Low Voltage (pins 6, 11)			–0.3		1.5	V
V_{en} = H	Enable High Voltage (pins 6, 11)			2.3		V_{SS}	V
I_{en} = L	Low Voltage Enable Current (pins 6, 11)	V_{en} = L				–10	µA
I_{en} = H	High Voltage Enable Current (pins 6, 11)	V_{en} = H ≤ V_{SS} –0.6V			30	100	µA
$V_{CE(sat)(H)}$	Source Saturation Voltage	I_L = 1A I_L = 2A		0.95	1.35 2	1.7 2.7	V V
$V_{CE(sat)(L)}$	Sink Saturation Voltage	I_L = 1A (5) I_L = 2A (5)		0.85	1.2 1.7	1.6 2.3	V V
$V_{CE(sat)}$	Total Drop	I_L = 1A (5) I_L = 2A (5)		1.80		3.2 4.9	V V
V_{sens}	Sensing Voltage (pins 1, 15)			–1 (1)		2	V

L298

ELECTRICAL CHARACTERISTICS (continued)

Symbol	Parameter	Test Conditions	Min.	Typ.	Max.	Unit
T_1 (V_i)	Source Current Turn-off Delay	0.5 V_i to 0.9 I_L (2); (4)		1.5		µs
T_2 (V_i)	Source Current Fall Time	0.9 I_L to 0.1 I_L (2); (4)		0.2		µs
T_3 (V_i)	Source Current Turn-on Delay	0.5 V_i to 0.1 I_L (2); (4)		2		µs
T_4 (V_i)	Source Current Rise Time	0.1 I_L to 0.9 I_L (2); (4)		0.7		µs
T_5 (V_i)	Sink Current Turn-off Delay	0.5 V_i to 0.9 I_L (3); (4)		0.7		µs
T_6 (V_i)	Sink Current Fall Time	0.9 I_L to 0.1 I_L (3); (4)		0.25		µs
T_7 (V_i)	Sink Current Turn-on Delay	0.5 V_i to 0.9 I_L (3); (4)		1.6		µs
T_8 (V_i)	Sink Current Rise Time	0.1 I_L to 0.9 I_L (3); (4)		0.2		µs
fc (V_i)	Commutation Frequency	I_L = 2A		25	40	KHz
T_1 (V_{en})	Source Current Turn-off Delay	0.5 V_{en} to 0.9 I_L (2); (4)		3		µs
T_2 (V_{en})	Source Current Fall Time	0.9 I_L to 0.1 I_L (2); (4)		1		µs
T_3 (V_{en})	Source Current Turn-on Delay	0.5 V_{en} to 0.1 I_L (2); (4)		0.3		µs
T_4 (V_{en})	Source Current Rise Time	0.1 I_L to 0.9 I_L (2); (4)		0.4		µs
T_5 (V_{en})	Sink Current Turn-off Delay	0.5 V_{en} to 0.9 I_L (3); (4)		2.2		µs
T_6 (V_{en})	Sink Current Fall Time	0.9 I_L to 0.1 I_L (3); (4)		0.35		µs
T_7 (V_{en})	Sink Current Turn-on Delay	0.5 V_{en} to 0.9 I_L (3); (4)		0.25		µs
T_8 (V_{en})	Sink Current Rise Time	0.1 I_L to 0.9 I_L (3); (4)		0.1		µs

1) 1)Sensing voltage can be –1 V for t ≤ 50 µsec; in steady state V_{sens} min ≥ –0.5 V.
2) See fig. 2.
3) See fig. 4.
4) The load must be a pure resistor.

16.3 CARACTERÍSTICAS TÉCNICAS DE ARDUINO UNO

Resumen de características Técnicas

Microcontrolador	Atmega328
Voltaje de operación	5V
Voltaje de entrada (Recomendado)	7 – 12V
Voltaje de entrada (Límite)	6 – 20V
Pines para entrada- salida digital.	14 (6 pueden usarse como salida de PWM)
Pines de entrada analógica.	6
Corriente continua por pin IO	40 mA
Corriente continua en el pin 3.3V	50 mA
Memoria Flash	32 KB (0,5 KB ocupados por el bootloader)
SRAM	2 KB
EEPROM	1 KB
Frecuencia de reloj	16 MHz

16.4 CÓDIGO DE ARDUINO PARA CALCULAR RPM

```
//Variables globales
int rpm;
volatile byte pulsos;
unsigned long timeold;
unsigned int pulsos_por_vuelta = 20;

//Interrupcion
void contador ()
{
  pulsos++;
}

void setup() {
  Serial.begin(115200);
  pinMode(2,INPUT);

  attachInterrupt(0,contador,RISING);
  pulsos = 0;
  rpm = 0;
  timeold = 0;
}

void loop() {
  if (millis() - timeold >= 1000)
  {
    detachInterrupt(0);
    rpm = (60 * 1000 / pulsos_por_vuelta)/(millis() - timeold) * pulsos;
    timeold = millis();
    pulsos = 0;

    Serial.print("RPM = ");
```

```
    Serial.println(rpm,DEC);


    attachInterrupt(0,contador,RISING);
  }
}
```

16.5 CÓDIGO DE ARDUINO PARA PRUEBA DE LA IMU

```
#include <Wire.h>

//Direccion I2C de la IMU
#define MPU 0x68

//Ratios de conversion
#define A_R 16384.0
#define G_R 131.0

//Conversion de radianes a grados 180/PI
#define RAD_A_DEG = 57.295779

//MPU-6050 da los valores en enteros de 16 bits
//Valores sin refinar
int16_t AcX, AcY, AcZ, GyX, GyY, GyZ;

//Angulos
float Acc[2];
float Gy[2];
float Angle[2];

void setup()
{
Wire.begin();
Wire.beginTransmission(MPU);
Wire.write(0x6B);
Wire.write(0);
Wire.endTransmission(true);
Serial.begin(9600);
}
```

```
void loop()
{
  //Leer los valores del Acelerometro de la IMU
  Wire.beginTransmission(MPU);
  Wire.write(0x3B); //Pedir el registro 0x3B - corresponde al AcX
  Wire.endTransmission(false);
  Wire.requestFrom(MPU,6,true); //A partir del 0x3B, se piden 6 registros
  AcX=Wire.read()<<8|Wire.read(); //Cada valor ocupa 2 registros
  AcY=Wire.read()<<8|Wire.read();
  AcZ=Wire.read()<<8|Wire.read();

   //A partir de los valores del acelerometro, se calculan los angulos Y, X
   //respectivamente, con la formula de la tangente.
  Acc[1] = atan(-1*(AcX/A_R)/sqrt(pow((AcY/A_R),2) +
pow((AcZ/A_R),2)))*RAD_TO_DEG;
  Acc[0] = atan((AcY/A_R)/sqrt(pow((AcX/A_R),2) +
pow((AcZ/A_R),2)))*RAD_TO_DEG;

  //Leer los valores del Giroscopio
  Wire.beginTransmission(MPU);
  Wire.write(0x43);
  Wire.endTransmission(false);
  Wire.requestFrom(MPU,4,true); //A diferencia del Acelerometro, solo se piden 4
registros
  GyX=Wire.read()<<8|Wire.read();
  GyY=Wire.read()<<8|Wire.read();

  //Calculo del angulo del Giroscopio
  Gy[0] = GyX/G_R;
  Gy[1] = GyY/G_R;

  //Aplicar el Filtro Complementario
  Angle[0] = 0.98 *(Angle[0]+Gy[0]*0.010) + 0.02*Acc[0];
```

```
  Angle[1] = 0.98 *(Angle[1]+Gy[1]*0.010) + 0.02*Acc[1];


  //Mostrar los valores por consola
  Serial.print("Angle X: "); Serial.print(Angle[0]); Serial.print("\n");
  Serial.print("Angle Y: "); Serial.print(Angle[1]); Serial.print("\n----------\n");


  delay(10); //Nuestra dt sera, pues, 0.010, que es el intervalo de tiempo en cada medida
}
```

16.6 CÓDIGO DE ARDUINO PARA PRUEBA DE LOS MOTORES DC

```
// Motor A
int in1 = 9;  // Pin que controla el sentido de giro
int in2 = 8;  // Pin que controla el sentido de giro

// Motor B
int enB = 5;  // Pin habilita motor B - PWM
int in3 = 7;  // Pin que controla el sentido de giro
int in4 = 6;  // Pin que controla el sentido de giro

void setup ()
{
  // Configura todos los pines como salida
  pinMode(enB, OUTPUT);
  pinMode(in1, OUTPUT);
  pinMode(in2, OUTPUT);
  pinMode(in3, OUTPUT);
  pinMode(in4, OUTPUT);
}

void VelocidadFija()
{
  // Gira el Motor A en ambas direcciones a una velocidad fija
  // Gira el motor A en sentido horario
  digitalWrite(in1, HIGH);
  digitalWrite(in2, LOW);
  delay(3000); // Giramos los motores por 3 segundos

  // Detenemos el Motor A
  digitalWrite(in1, LOW);
  digitalWrite(in2, LOW);
```

```
}

void VelocidadVariable()
{
  // Giramos el Motor B en un rango de velocidad
  // Usamos PWM para enviar analogWrite() y controlar la maxima velocidad posible

  // Giramos los motores en un sentido
  digitalWrite(in3, LOW);
  digitalWrite(in4, HIGH);

  // Aceleramos desde 0 hasta la velocidad maxima
  for(int i = 0; i < 255; i++)
  {
    analogWrite(enB, i); // Usamos PWM para variar la velocidad
    Serial.print("Velocidad: ");
    Serial.println(i);
    delay(20);
  }

  // Desaceleramos desde la velocidad maxima hasta llegar a 0
  for(int i = 255; i > 0; i--)
  {
    analogWrite(enB, i); // Usamos PWM para variar la velocidad
    Serial.print("Velocidad: ");
    Serial.println(i);
    delay(20);
  }

  // Detenemos los motores
  digitalWrite(in3, LOW);
  digitalWrite(in4, LOW);
```

```
}



void loop()
{
  Serial.begin (9600);       // inicializa el puerto seria a 9600 baudios
  VelocidadFija();
  Serial.println("Inicia Prueba de Velocidad Fija");
  delay(2000);
  VelocidadVariable();
  Serial.println("Inicia Prueba de Velocidad Variable");
  delay(2000);
}
```

16.7 CÓDIGO DE ARDUINO PRINCIPAL

```
//Libreria necesaria
#include <Wire.h>

//Direccion I2C de la IMU
#define MPUAddress 0x68

//Conversion de radianes a grados 180/pi
#define RadAGrados 57.295779

//Ratios de conversion del acelerometro y giroscopio
#define AR 16384.0
#define GR 131.0

//La IMU MPU-6050 da los valores en enteros de 16 bits
int16_t AX, AY, AZ, GX, GY, GZ;

//Angulos necesarios
float Ac;
float Gy;
float Angulo;

//Pines del driver
int IN1 = 9;
int IN2 = 10;
int IN3 = 6;
int IN4 = 5;
int SalI, SalD;

//Configuracion del PID
unsigned long lastTime;
double Input, Output, Setpoint, output;
```

```
double Ierror, lastErr;

double kp = 20;
double ki = 80;
double kd = 0.5;

int SampleTime = 10; //10 ms
double SampleTimeSec = (double)SampleTime / 1000;
double outMin = -255;
double outMax = 255;

void Compute()
{
  unsigned long now = millis();
  int timeChange = (now - lastTime);
  //Serial.println(timeChange);
  if (timeChange >= SampleTime)
  {
    //Errores
    double error = Setpoint - Input;
    Ierror += error * SampleTimeSec;
    double dErr = (error - lastErr) / SampleTimeSec;

    //Salida
    Output = kp * error + ki * Ierror + kd * dErr;

    if (Output > outMax)
    {
      Output = outMax;
      //Ierror = 0;
    }
    else if (Output < outMin)
```

```
    {
        Output = outMin;
        //Ierror = 0;
    }
    lastErr = error;
    lastTime = now;
  }
}

void setup()
{
Wire.begin();
Wire.beginTransmission(MPUAddress);
Wire.write(0x6B);
Wire.write(0);
Wire.endTransmission(true);
Serial.begin(9600);

//Configuracion pines del driver
pinMode (IN1,OUTPUT);
pinMode (IN2,OUTPUT);
pinMode (IN3,OUTPUT);
pinMode (IN4,OUTPUT);

//Configuracion del PID
Setpoint = 0;
Input = Angulo;
}

void loop()
{
  //Leer los valores del Acelerometro de la IMU
```

```
  Wire.beginTransmission(MPUAddress);
  Wire.write(0x3B); //Pedir el registro 0x3B que corresponde al AX
  Wire.endTransmission(false);
  Wire.requestFrom(MPUAddress,6,true); //A partir del 0x3B, se piden 6 registros

  AX = Wire.read()<<8|Wire.read(); //Cada valor ocupa 2 registros
  AY = Wire.read()<<8|Wire.read();
  AZ = Wire.read()<<8|Wire.read();

  //A partir de los valores del acelerometro, se calcula el angulo
  //con la formula de la tangente
  Ac = atan(-1*(AX/AR)/sqrt(pow((AY/AR),2) + pow((AZ/AR),2)))*RadAGrados;

  //Leer los valores del Giroscopio
  Wire.beginTransmission(MPUAddress);
  Wire.write(0x43); //Pedir el registro 0x43 que corresponde al GX
  Wire.endTransmission(false);
  Wire.requestFrom(MPUAddress,4,true); //A diferencia del Acelerometro, solo se
piden 4 registros

  GX = Wire.read()<<8|Wire.read();  //Cada valor ocupa 2 registros
  GY = Wire.read()<<8|Wire.read();

  //Calculo del angulo del Giroscopio
  Gy = GY / GR;

  //Se aplica el Filtro Complementario
  Angulo = 0.98 * (Angulo + Gy * 0.010) + 0.02 * Ac;

  //Configuracion lazo PID
  Input = Angulo;
  Compute(); //Funcion que calcula la salida PWM
  if (Output > 0)
```

```
  {
    SalI = 5;
    SalD = 9;
    digitalWrite(6,LOW);
    digitalWrite(10,LOW);
  }
  if (Output < 0)
  {
    SalI = 6;
    SalD = 10;
    digitalWrite(5,LOW);
    digitalWrite(9,LOW);
  }
  output = abs(Output);
  analogWrite(SalI,output);
  analogWrite(SalD,output);

  //Mostrar los valores por consola
  Serial.println (Angulo);
  //Serial.print("Salida controlador: "); Serial.println(output);
}
```

16.8 CÓDIGO DE MATLAB

```
%% Variables

g = 9.81;    %gravedad
H = 0.16;    %altura del pendulo
w = 0.08;    %anchura del pendulo
L = 0.07;    %distancia al centro de masas
r = 0.03;    %radio de la rueda
mp = 0.548;    %masa del pendulo
mr = 0.083;    %masa de las ruedas
Jp = 1/12*mp*H^2+1/12*mp*w^2+mp*L^2;    %inercia del pendulo
Jr = 1/2*mr*r^2;    %inercia de las ruedas
Ra = 7;    %resistencia de armadura
Kt = 0.08;  %kt=ke
Ke = 0.08;  %ke=fem/w
n = 1;    %relacion de transmisión

%% Ecuaciones

a1 = (g*L^2*r*mp^2)/(-L^2*r^2*mp^2+Jp*(Jr+r^2*(mp+mr)));

a2 = (-2*n*Jp*Ke*Kt-2*L*n*r*Ke*Kt*mp)/((-
L^2*r^2*mp^2+Jp*(Jr+r^2*(mp+mr)))*Ra);

a3 = (g*L*Jr*mp*Ra+L*r^2*mp*(g*mp+g*mr)*Ra)/((-
L^2*r^2*mp^2+Jp*(Jr+r^2*(mp+mr)))*Ra);

a4 = (-2*n*Jr*Ke*Kt-2*n*r*Ke*Kt*((L+r)*mp+r*mr))/((-
L^2*r^2*mp^2+Jp*(Jr+r^2*(mp+mr)))*Ra);

b1 = (2*n*Jp*Kt+2*L*n*r*Kt*mp)/((-
L^2*r^2*mp^2+Jp*(Jr+r^2*(mp+mr)))*Ra);

b2 = (2*n*Jr*Kt+2*n*r*Kt*((L+r)*mp+r*mr))/((-
L^2*r^2*mp^2+Jp*(Jr+r^2*(mp+mr)))*Ra);

%% Modelo Planta

A = [0 0 1;a1 a2 0;a3 a4 0];
B = [0;b1;b2];
C = [1 0 0];
D = [0];

%% Funcion de transferencia

[num,den] = ss2tf(A,B,C,D)
Gs = tf(num,den)

%% Sisotool

Ps = zpk(Gs)*9/255*180/pi
```

```
Kd = 0.5
Kp = 20
Ki = 80
Cs = (Kd*s^2+Kp*s+Ki)/s
sisotool(Ps,Cs)
```

16.9 TIPOS DE ENCODERS

16.9.1 ENCODER LINEAL

Es el tipo de encoder más sencillo y se utiliza por ejemplo en los calibres digitales, en donde la medida se muestra en un display digital. Detrás de la pieza móvil hay un encoder capacitivo que se encarga de leer la distancia recorrida desde el cero, por interpolación.

Este tipo de encoder se compone de un módulo fijo y otro móvil. El módulo fijo contiene el sensor y la electrónica necesarias para detectar y medir el movimiento, y convertirlo en impulsos eléctricos inteligibles por otro circuito digital o analógico.

Figura 58. Encoder lineal

16.9.2 ENCODER ROTATORIO

En este tipo de encoder la lectura se realiza sobre un disco, en el cual se encuentra la codificación que permite diferenciar la posición angular con gran precisión.

Las aplicaciones más comunes se dan en controles de máquinas industriales como por ejemplo los husillos de tornos, las fresadoras, los brazos robóticos… Este tipo de encoder se usaba también en los ratones viejos de ordenadores.

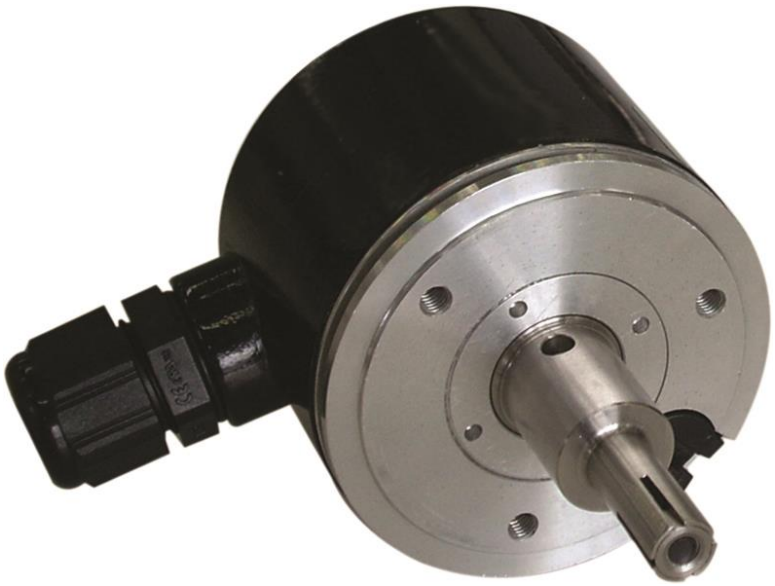


Figura 59. Encoder rotatorio

16.9.3 ENCODER INCREMENTAL

Este tipo de sensor detecta el movimiento y la distancia recorrida en ese movimiento debido a la detección diferencial de dos valores codificados en la superficie detectable.

Como ventaja tiene que es más económico que el encoder absoluto y permite mantener la misma precisión sin importar la longitud de la pieza móvil. La desventaja que tiene es que el sistema necesita posicionar el encoder en un cero predeterminado para poder inicializarse.

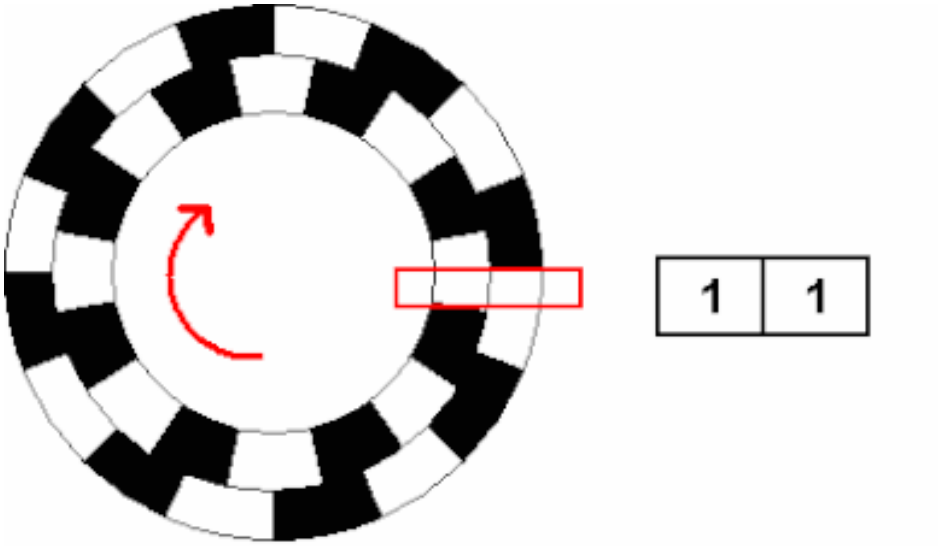


Figura 60. Encoder incremental

16.9.4 ENCODER ABSOLUTO

En este encoder la codificación sobre la superficie de la pieza móvil incluye la posición real desde el punto cero de la escala. De esta manera, el sensor puede conocer su posición sin necesidad de moverlo.

La desventaja principal es que se requieren más pistas de codificación para incluir la información de posición, lo cual encarece la electrónica necesaria.

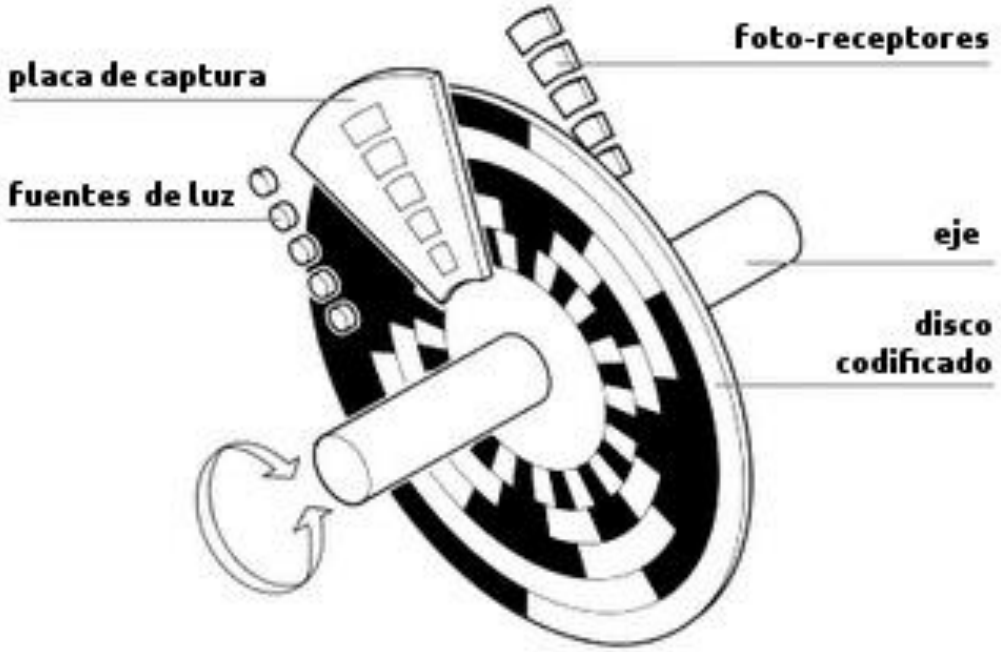


Figura 61. Encoder absoluto

16.9.5 ENCODER LINEAL MAGNÉTICO

Estos encoders usan una cinta magnetizada o una fabricada con un material de reluctancia magnética variable. La posición se calcula a través de cabezales de lectura magneto-resistivos o con solenoides.

El problema principal de estos sensores es que son susceptibles a los campos magnéticos intensos, por lo tanto, no son recomendables en lugares donde existan fuerzas electromotrices elevadas, como por ejemplo transformadores de alta tensión o grandes motores.

La ventaja que tienen estos encoders es que son muy fiables en la lectura, relativamente económicos y pueden alcanzar resoluciones del orden de los micrómetros.

Figura 62. Encoder lineal magnético

16.9.6 ENCODER LINEAL ÓPTICO

Estos encoders registran los cambios en una cinta, los cuales pueden estar codificados en patrones Moiré, holográficos u otros.

La ventaja principal de este tipo de encoders es la gran precisión que permiten alcanzar, del orden de décimas de micrón. También pueden funcionar sin la necesidad de contacto físico entre las partes por lo que es apropiado para aplicaciones en las que el rozamiento debe evitarse.

Por otra parte, la desventaja principal es su gran susceptibilidad a la suciedad, por lo que es necesario que el conjunto esté hermético.

Figura 63. Encoder lineal óptico

16.9.7 ENCODER ROTATIVO CAPACITIVO

Este tipo de encoder funciona midiendo la capacitancia entre la escala y el cabezal lector. Como la lectura se realiza sin contacto físico se utiliza principalmente en aplicaciones de medición como por ejemplo calibres, diales…

La desventaja principal de este encoder es que es susceptible a la presencia de suciedad en el cabezal lector o en la escala, por lo tanto, es necesario que disponga de un cierre hermético.

Figura 64. Encoder rotativo capacitivo

16.9.8 ENCODER INDUCTIVO

Este sensor es el más robusto y se puede usar en ambientes en donde hay contaminantes como por ejemplo líquidos refrigerantes o partículas. Como principal desventaja es que no tiene tanta precisión como los casos anteriores.

Figura 65. Encoder inductivo

PLANOS

TRABAJO FIN DE GRADO

UNIVERSIDAD DE LA RIOJA

GRADO EN INGENIERÍA ELECTRÓNICA INDUSTRIAL Y AUTOMÁTICA

CURSO 2018/19

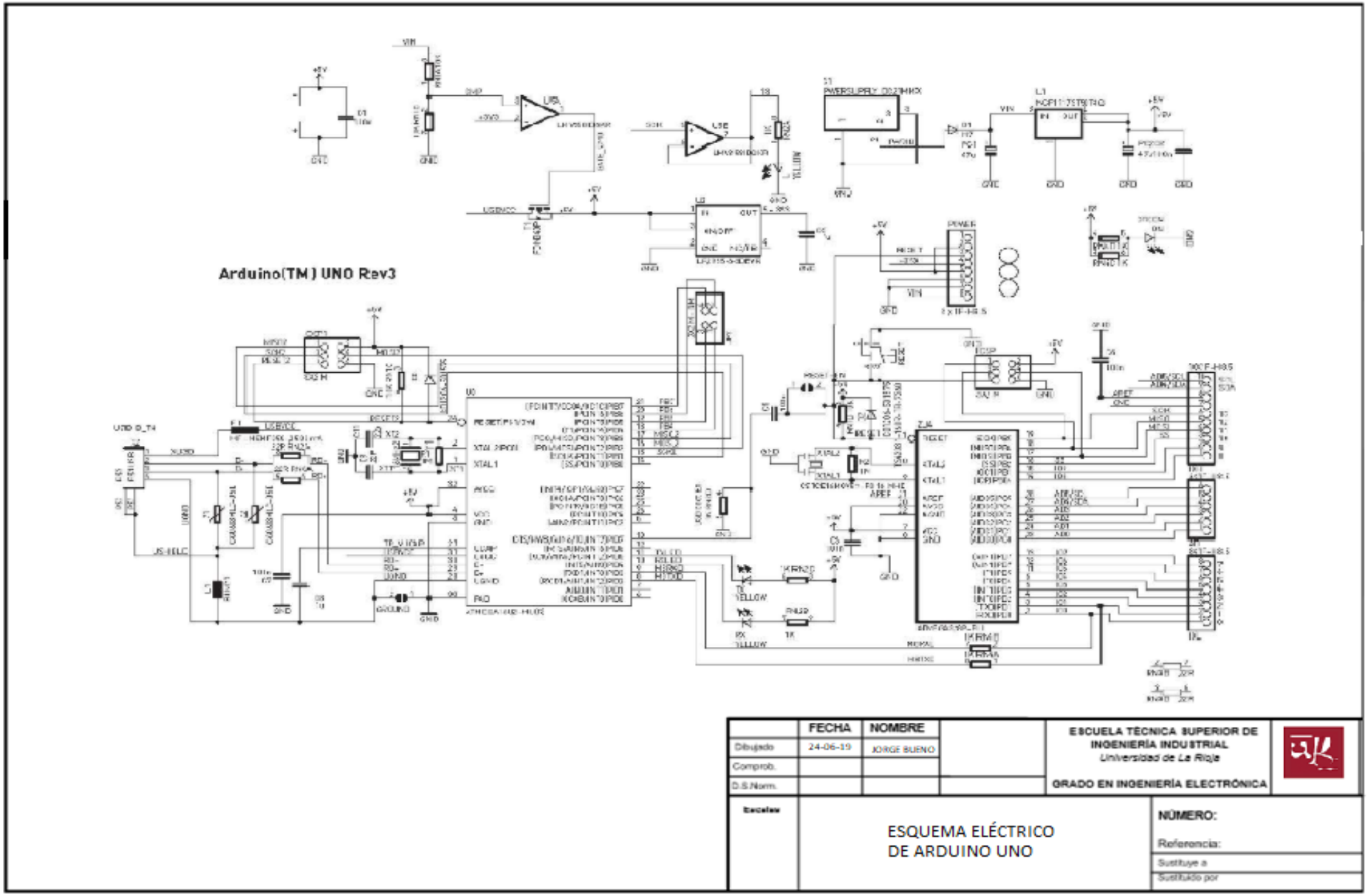
REALIZADO POR: JORGE BUENO GIL

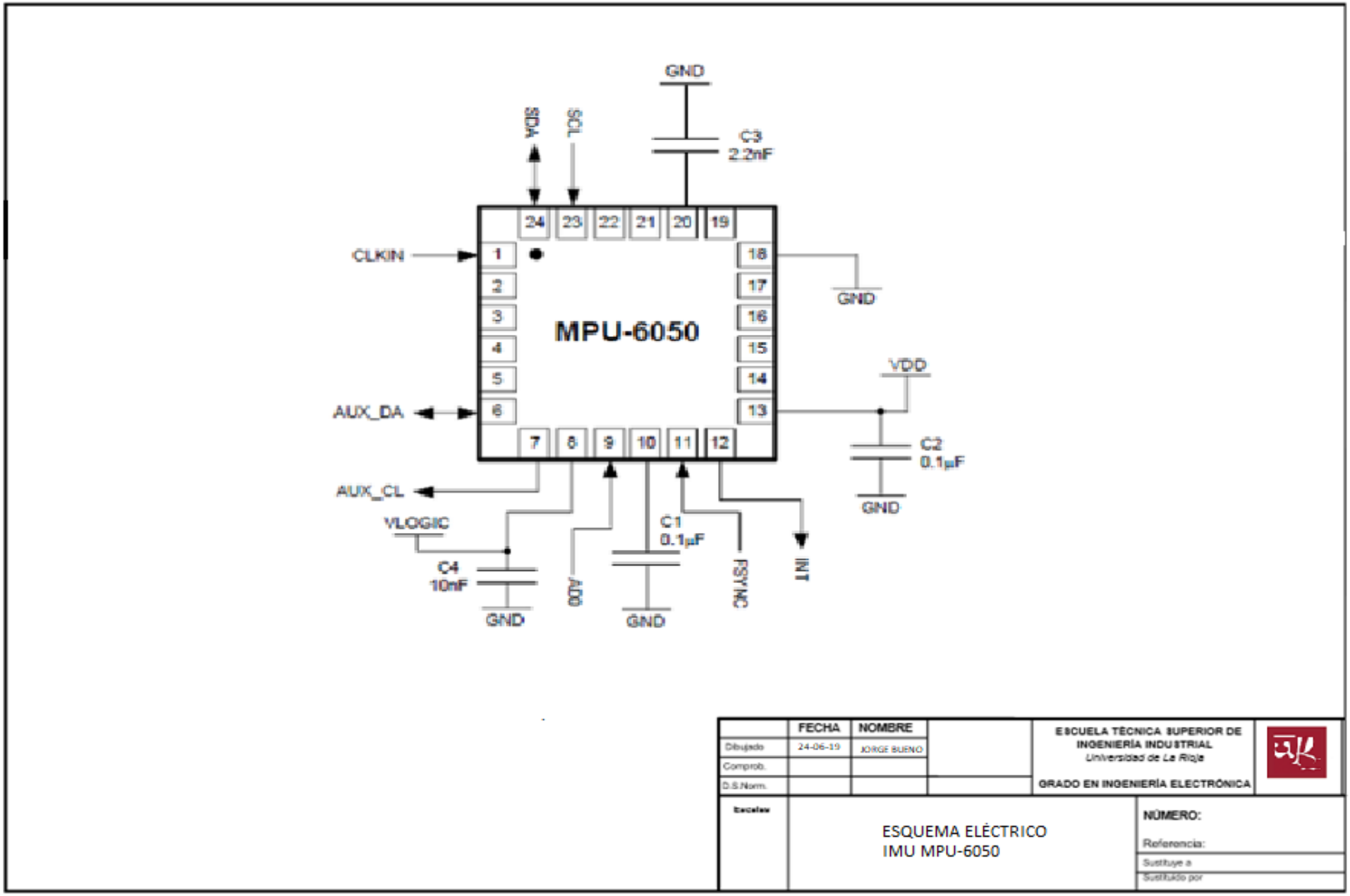
TUTORES DEL PROYECTO: JAVIER RICO AZAGRA
MONTSERRAT GIL MARTÍNEZ

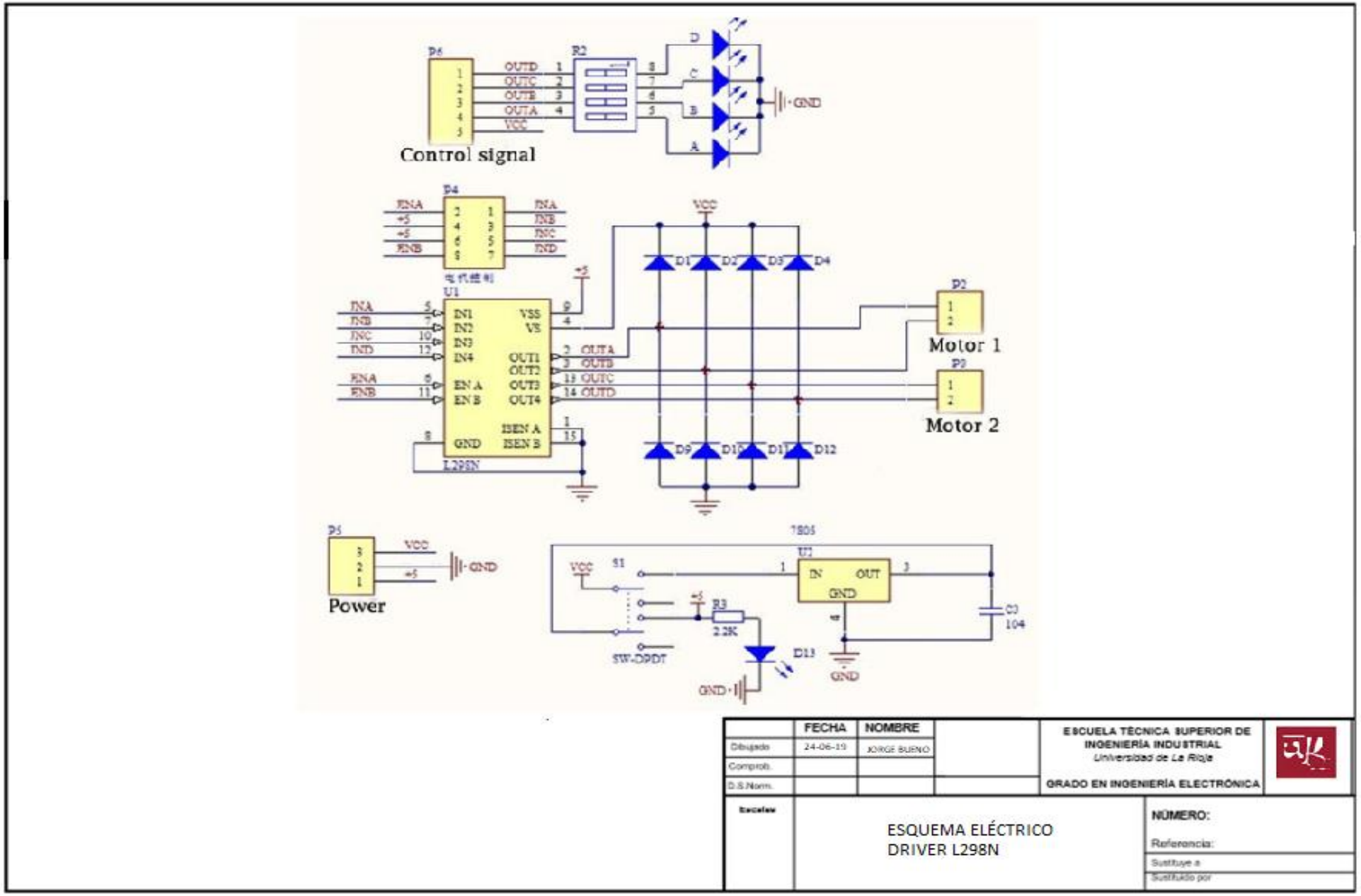
ÍNDICE PLANOS

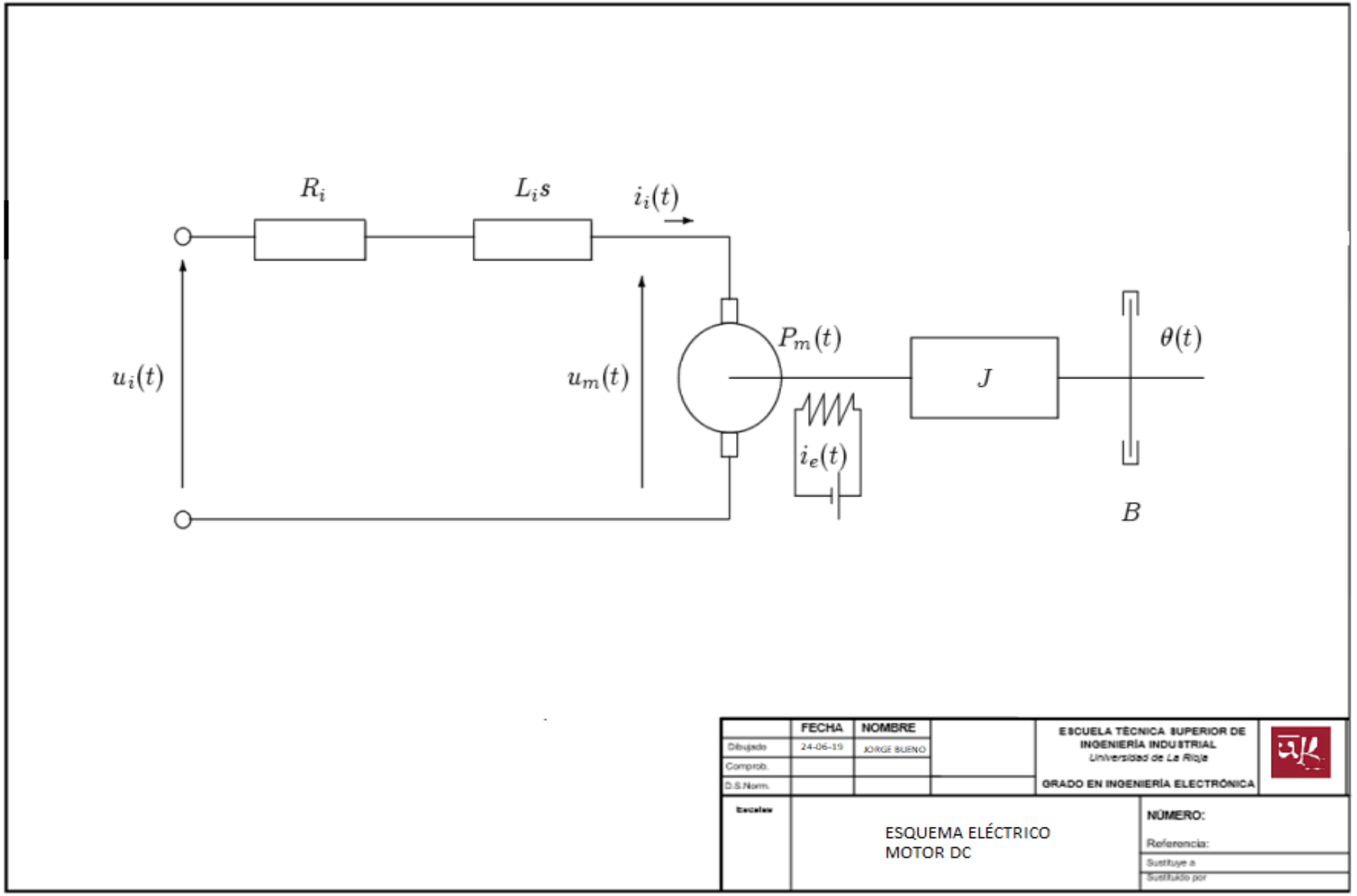
17 PLANOS

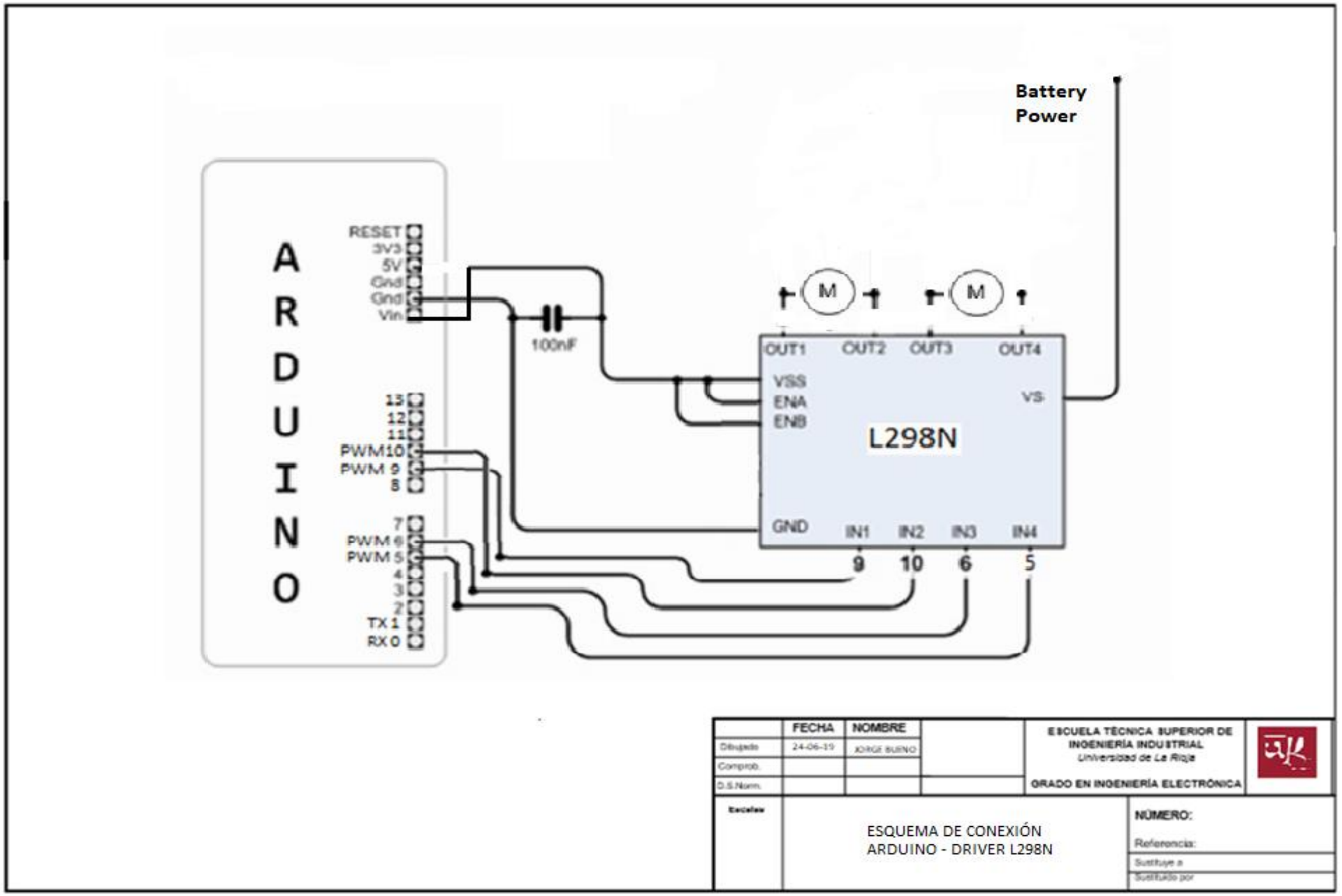
En las siguientes páginas de este documento se muestran los esquemas eléctricos de los diferentes dispositivos del proyecto y los esquemas de conexiones entre ellos necesarios para que funcione el sistema.

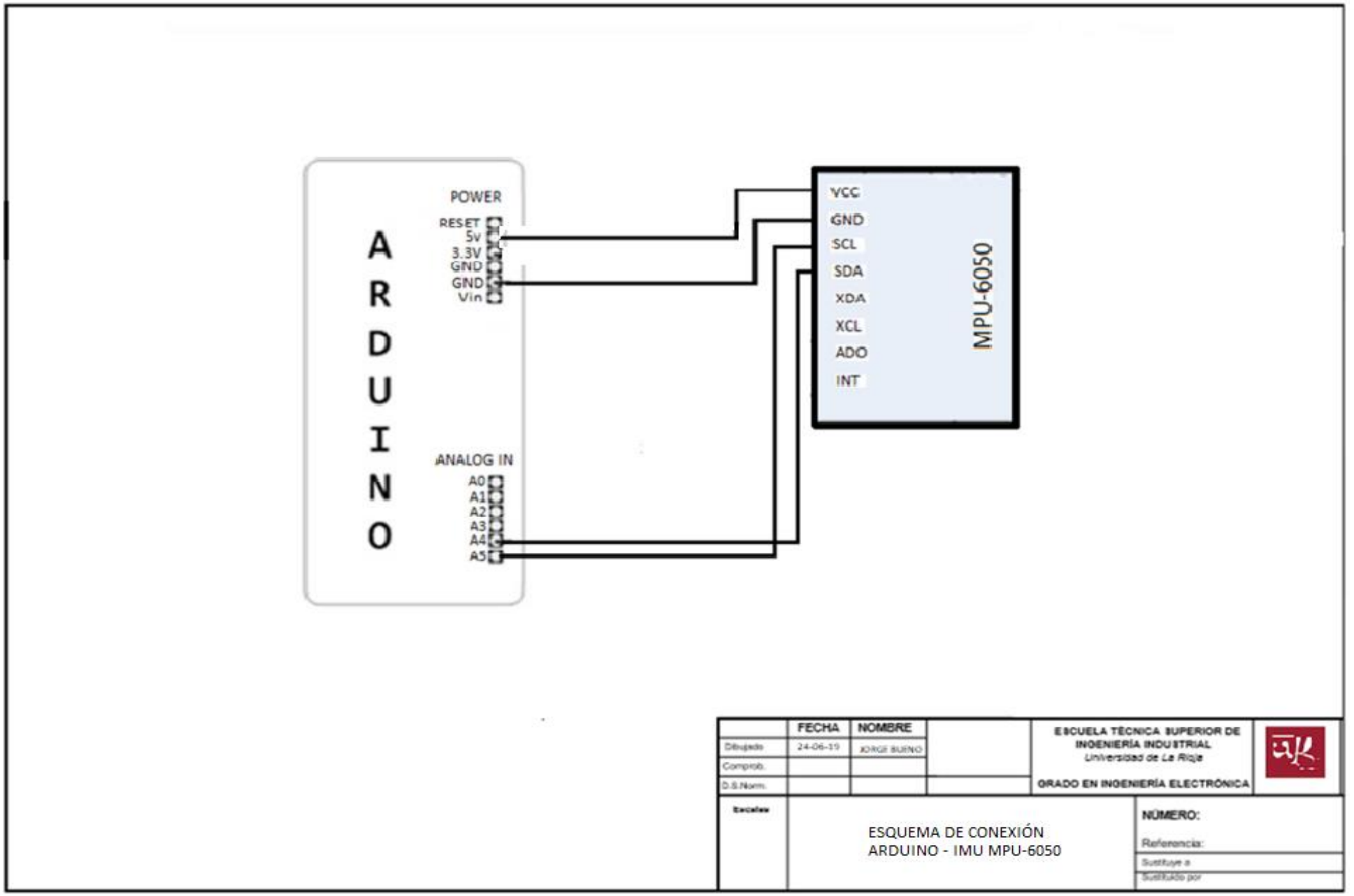

Arduino(TM) UNO Rev3
FECHA
NOMBRE
Dibujado
24-06-19
JORGE BUENO
Comprob.
D.S.Norm.
ESCUELA TÉCNICA SUPERIOR DE INGENIERÍA INDUSTRIAL
Universidad de La Rioja
GRADO EN INGENIERÍA ELECTRÓNICA
Escalas
ESQUEMA ELÉCTRICO DE ARDUINO UNO
NÚMERO:
Referencia:
Sustituye a
Sustituido por


GND
SDA
SCL
C3
2.2nF
24
23
22
21
20
19
CLKIN
1
2
3
4
5
6
MPU-6050
18
17
16
15
14
13
GND
VDD
C2
0.1µF
GND
AUX_DA
AUX_CL
7
8
9
10
11
12
VLOGIC
C4
10nF
GND
ADO
C1
0.1µF
GND
FSYNC
INT
FECHA
NOMBRE
Dibujado
24-06-19
JORGE BUENO
Comprob.
D.S.Norm.
ESCUELA TÉCNICA SUPERIOR DE INGENIERÍA INDUSTRIAL
Universidad de La Rioja
GRADO EN INGENIERÍA ELECTRÓNICA
Escalas
ESQUEMA ELÉCTRICO
IMU MPU-6050
NÚMERO:
Referencia:
Sustituye a
Sustituido por


Control signal
Motor 1
Motor 2
Power
L298N
7805
SW-DPDT
GND
VCC
ESCUELA TÉCNICA SUPERIOR DE INGENIERÍA INDUSTRIAL
Universidad de La Rioja
GRADO EN INGENIERÍA ELECTRÓNICA
FECHA
NOMBRE
24-06-19
JORGE BUENO
ESQUEMA ELÉCTRICO
DRIVER L298N
NÚMERO:
Referencia:


R_i
$L_i s$
$i_i(t)$
$u_i(t)$
$u_m(t)$
$P_m(t)$
J
$\theta(t)$
$i_e(t)$
B
FECHA
NOMBRE
Dibujado
24-06-19
JORGE BUENO
Comprob.
D.S.Norm.
ESCUELA TÉCNICA SUPERIOR DE INGENIERÍA INDUSTRIAL
Universidad de La Rioja
GRADO EN INGENIERÍA ELECTRÓNICA
Escalas
ESQUEMA ELÉCTRICO
MOTOR DC
NÚMERO:
Referencia:
Sustituye a
Sustituido por


Battery
Power
ARDUINO
RESET
3V3
5V
Gnd
Gnd
Vin
100nF
13
12
11
PWM10
PWM 9
8
7
PWM 6
PWM 5
4
3
2
TX 1
RX 0
M
M
OUT1
OUT2
OUT3
OUT4
VSS
ENA
ENB
VS
L298N
GND
IN1
IN2
IN3
IN4
9
10
6
5
FECHA
NOMBRE
Dibujado
24-06-19
JORGE BURNO
Comprob.
D.S.Norm.
ESCUELA TÉCNICA SUPERIOR DE INGENIERÍA INDUSTRIAL
Universidad de La Rioja
GRADO EN INGENIERÍA ELECTRÓNICA
ESQUEMA DE CONEXIÓN
ARDUINO - DRIVER L298N
NÚMERO:
Referencia:
Sustituye a
Sustituido por


ARDUINO
POWER
RESET
5V
3.3V
GND
GND
Vin
ANALOG IN
A0
A1
A2
A3
A4
A5
VCC
GND
SCL
SDA
XDA
XCL
ADO
INT
MPU-6050
FECHA
NOMBRE
Dibujado
24-06-19
JORGE BUENO
Comprob.
D.S.Norm.
ESCUELA TÉCNICA SUPERIOR DE INGENIERÍA INDUSTRIAL
Universidad de La Rioja
GRADO EN INGENIERÍA ELECTRÓNICA
Escalas
ESQUEMA DE CONEXIÓN
ARDUINO - IMU MPU-6050
NÚMERO:
Referencia:
Sustituye a
Sustituido por

PLIEGO DE CONDICIONES

TRABAJO FIN DE GRADO

UNIVERSIDAD DE LA RIOJA

GRADO EN INGENIERÍA ELECTRÓNICA INDUSTRIAL Y AUTOMÁTICA

CURSO 2018/19

REALIZADO POR: JORGE BUENO GIL

TUTORES DEL PROYECTO: JAVIER RICO AZAGRA
MONTSERRAT GIL MARTÍNEZ

ÍNDICE PLIEGO DE CONDICIONES

18 PLIEGO DE CONDICIONES

18.1 INTRODUCCIÓN

El autor de este trabajo ha cursado los estudios de Grado en Ingeniería Electrónica Industrial y Automática en la Universidad de La Rioja, cumpliendo con la normativa establecida por la Escuela Superior de Ingeniería Industrial en la normativa de Trabajo Fin de Grado.

El objetivo de este pliego es reunir y establecer todas las disposiciones técnicas, administrativas y económicas, y las normativas que ha de regir este proyecto.

El diseño del proyecto y sus características han sido descritos detalladamente en la memoria del proyecto y en los anexos.

Las condiciones que se precisan en este documento tratan de cumplir con la calidad esperada para este proyecto. Si el proyecto no se realiza según estas condiciones, el proyectista no se hará cargo de los fallos o averías que puedan surgir en su funcionamiento, y los problemas derivados afectarán a terceras personas.

Todos los cambios y modificaciones que pueda sufrir el proyecto y sus documentos deberán ser aprobados por el ingeniero o proyectista.

18.2 CONDICIONES GENERALES

Este proyecto se adecúa a los reglamentos y normativas electrónicas vigentes. Cuando el proyecto esté terminado se podrán realizar modificaciones, pero siempre con la supervisión del proyectista.

La propiedad intelectual del autor y director del Trabajo Fin de Grado se regirá por el Real Decreto Legislativo 1/1996, de 12 de abril, por el cual se aprueba el texto refundido de la Ley de Propiedad Intelectual.

18.3 CONDICIONES ADMINISTRATIVAS

El proyecto consta de los siguientes documentos:

- Índice General: indica la página de comienzo de cada uno de los apartados que forman el proyecto.
- Memoria: en ella se consideran las necesidades a satisfacer y los factores técnicos a tener en cuenta explicando las posibles soluciones técnicas y justificando la solución elegida.
- Anexos: se recoge la documentación considerada de interés para poder ampliar la descripción de los componentes del sistema, así como los códigos de los programas utilizados.
- Planos: se muestran los esquemas eléctricos de los dispositivos utilizados y los esquemas de conexiones entre los distintos componentes electrónicos.
- Pliego de Condiciones: se establecen las condiciones técnicas, administrativas y económicas para que el proyecto pueda llevarse a cabo, evitando posibles malinterpretaciones.
- Mediciones: se recogen las unidades necesarias de todos los componentes empleados en el proyecto.
- Presupuesto: se recoge el coste de todos los componentes empleados, el de la mano de obra y el de las licencias de software. Sumando todos estos costes se obtiene el coste final del proyecto.

18.4 NORMATIVA Y REGLAMENTACIÓN

Este proyecto está regido tanto por la normativa española como por la normativa internacional.

18.4.1 NORMATIVA RELACIONADA CON PRODUCTOS ELECTRÓNICOS

En relación al desarrollo de productos electrónicos, en AENOR (Asociación Española de Normalización y Certificación) se pueden encontrar las siguientes normativas:

- Norma UNE1302—2:1973. Vocabulario electrotécnico. Electrónica.

- Norma UNE-EN611000-4-3-1998. Compatibilidad electromagnética.

Este proyecto se rige también bajo el reglamento de Baja Tensión.

18.4.2 REGLAMENTO RELACIONADO CON MATERIALES Y EQUIPOS

Los materiales y equipos de este proyecto deben cumplir con los estándares nacionales e internacionales en vigor y de obligado cumplimiento. Entre otros:

- UNE 20-324. Grados de protección de los envolventes del material eléctrico de baja tensión.
- UNE 20-334. Conductos para instalaciones eléctricas.
- UNE 21-401. Conductores eléctricos aislados.
- UNE 21-402. Conductores eléctricos aislados y desnudos.

18.5 CONDICIONES FACULTATIVAS

18.5.1 DIRECCIÓN

La dirección del montaje será realizada por el ingeniero proyectista o por cualquier otra persona en la que este delegue, teniendo en cuenta la capacidad de dicha persona para hacerse cargo de dicha dirección.

Cuando el proyecto esté terminado, este podrá ser usado por cualquier persona con conocimientos suficientes sobre el sistema, sus componentes y el funcionamiento global del mismo.

Si ocurre alguna avería o pérdida de información debido a una utilización incorrecta, el ingeniero proyectista o la persona en cual haya delegado la dirección del proyecto, quedan exentas de responsabilidad.

18.5.2 LIBRO DE ÓRDENES

La instalación y montaje de todos los componentes necesarios del proyecto se realizará según el siguiente orden de prioridad si ocurre alguna contradicción:

- Presupuesto

- Pliego de condiciones
- Memoria

Este libro de órdenes debe estar conforme al Decreto 462/1971 de 11 de marzo, y la Orden de 9 de junio de 1971.

18.5.3 MODIFICACIONES

En caso de ser necesario tener que realizar alguna modificación en el presente proyecto, esta deberá ser comunicada con anterioridad al Ingeniero Director, quien deberá dar la correspondiente autorización.

Si se realizan modificaciones que no han sido previamente autorizadas por el Ingeniero Director, las consecuencias que dichos cambios puedan ocasionar serán de total responsabilidad de la persona que las haya realizado.

18.6 CONDICIONES DE MATERIALES Y EQUIPOS

En este apartado se detallan las condiciones tanto de hardware como de software que hay que tener en cuenta para poder utilizar la aplicación final del proyecto.

Todos los materiales y componentes empleados en el proyecto deben cumplir con las especificaciones técnicas que aparecen descritas en la memoria y deben cumplir también todas las normas descritas en este pliego de condiciones.

Si fuera necesario reemplazar algún componente o material por otro nuevo, estos deberán tener las mismas características que los reemplazados, inhibiéndose el proyectista de cualquier responsabilidad por fallo si no se cumplen estos requisitos.

Para la implementación y el desarrollo del proyecto se deberá disponer de un ordenador, el cual deberá tener instalado al menos el siguiente software:

- Windows 10
- Matlab – Simulink R2018b
- Arduino 1.8.9

18.7 CONDICIONES ECONÓMICAS

18.7.1 ERRORES EN EL PROYECTO

Si existe algún tipo de error en el proyecto se avisará inmediatamente al proyectista y se le informará con detalle de los errores encontrados.

Además, se dejará de utilizar el robot equilibrista hasta que los errores queden totalmente subsanados para prevenir cualquier tipo de daño.

18.7.2 LIQUIDACIÓN

Una vez que se haya terminado la realización del proyecto, se procederá a la liquidación final, en la que se incluye el importe de las unidades de realización, además de las posibles modificaciones del proyecto que hayan sido aprobadas por la dirección del proyecto.

Cuando se suscriba el contrato, el cliente habrá de abonar el 80% del presupuesto. El 20% quedará como garantía durante los seis primeros meses, a partir de la fecha de puesta en marcha del sistema.

Si transcurren los seis meses y no se ha manifestado ningún defecto o error de funcionamiento, el cliente abonará el 20% que estaba pendiente. A partir de ese momento, se considerarán concluidos los compromisos entre ambas partes, excepto el periodo de garantía.

18.8 DISPOSICIÓN FINAL

Las partes contratantes, tanto la dirección del proyecto como el cliente, se ratifican en el contenido del presente pliego de condiciones, que tiene igual validez, a todos los efectos, que una escritura pública, prometiendo su fiel cumplimiento.

MEDICIONES

TRABAJO FIN DE GRADO

UNIVERSIDAD DE LA RIOJA

GRADO EN INGENIERÍA ELECTRÓNICA INDUSTRIAL Y AUTOMÁTICA

CURSO 2018/19

REALIZADO POR: JORGE BUENO GIL

TUTORES DEL PROYECTO: JAVIER RICO AZAGRA
MONTSERRAT GIL MARTÍNEZ

ÍNDICE MEDICIONES

19 ESTADO DE MEDICIONES

En este apartado se presentan las unidades necesarias de cada componente del proyecto.

COMPONENTES	UNIDADES
Tablerillo de madera 0.07x0.14m	2
Tablerillo de madera 0.07x0.11m	1
Tablerillo de madera 0.07x0.10m	1
Varilla roscada Diámetro 6mm	4
Varilla roscada Diámetro 3mm	2
Rueda Diámetro 6cm	2
Tuerca hexagonal Diámetro 6mm	16
Tuerca hexagonal Diámetro 3mm	4
Tuerca de seguridad Diámetro 6mm	12
Tuerca de seguridad Diámetro 3mm	4
Arandela Diámetro 6mm	16
Arandela Diámetro 3mm	8
Pletina Diámetro 3mm	2
Tornillo Diámetro 3mm	4
Brida	9
Cable Arduino macho-macho	2
Cable Arduino macho-hembra	8
Cable motor driver	4
IMU MPU-6050	1
Driver L298N	1
Arduino UNO	1
Pila AA 1.5V	6
Porta pilas	1

Motor DC	2
Interruptor	1

Tabla 2. Estado de mediciones

PRESUPUESTO

TRABAJO FIN DE GRADO

UNIVERSIDAD DE LA RIOJA

GRADO EN INGENIERÍA ELECTRÓNICA INDUSTRIAL Y AUTOMÁTICA

CURSO 2018/19

REALIZADO POR: JORGE BUENO GIL

TUTORES DEL PROYECTO: JAVIER RICO AZAGRA
MONTSERRAT GIL MARTÍNEZ

ÍNDICE PRESUPUESTO

20 DESARROLLO DEL PRESUPUESTO

A continuación, se presenta el coste de todos los componentes tanto mecánicos como eléctricos necesarios para la realización de este proyecto, así como el coste de la mano de obra y de las licencias de software necesarias.

Los costes están expresados en euros (€) y las cantidades económicas se expresan con dos decimales redondeando su valor cuando sea necesario.

20.1 CUADRO DE PRECIOS UNITARIOS TOTALES

En la siguiente tabla se presentan todos los componentes y dispositivos que se han utilizado en la realización de este proyecto con sus respectivos precios unitarios totales.

COMPONENTES	UNIDADES	PRECIO UNITARO TOTAL EN CIFRA (€)	PRECIO UNITARIO TOTAL EN LETRA (EUROS)
Tablerillo de madera 0.07x0.14m	2	1,90	Un euro con noventa céntimos
Tablerillo de madera 0.07x0.11m	1	0,65	Cero euros con sesenta y cinco céntimos
Tablerillo de madera 0.07x0.10m	1	0,55	Cero euros con cincuenta y cinco céntimos
Varilla roscada Diámetro 6mm	4	4,20	Cuatro euros con veinte céntimos
Varilla roscada Diámetro 3mm	2	1,30	Un euro con treinta céntimos
Rueda Diámetro 6cm	2	11,00	Once euros
Tuerca hexagonal Diámetro 6mm	16	3,20	Tres euros con veinte céntimos
Tuerca hexagonal Diámetro 3mm	4	0,48	Cero euros con cuarenta y ocho céntimos
Tuerca de seguridad Diámetro 6mm	12	3,36	Tres euros con treinta y seis céntimos
Tuerca de seguridad Diámetro 3mm	4	0,76	Cero euros con setenta y seis céntimos
Arandela Diámetro 6mm	16	2,40	Dos euros con cuarenta céntimos

Arandela Diámetro 3mm	8	0,64	Cero euros con sesenta y cuatro céntimos
Pletina Diámetro 3mm	2	1,30	Un euro con treinta céntimos
Tornillo Diámetro 3mm	4	1,40	Un euro con cuarenta céntimos
Brida	9	0,45	Cero euros con cuarenta y cinco céntimos
Cable Arduino macho-macho	2	0,16	Cero euros con dieciséis céntimos
Cable Arduino macho-hembra	8	0,96	Cero euros con noventa y seis céntimos
Cable motor driver	4	0,28	Cero euros con veintiocho céntimos
IMU MPU-6050	1	3,95	Tres euros con noventa y cinco céntimos
Driver L298N	1	5,95	Cinco euros con noventa y cinco céntimos
Arduino UNO	1	12,00	Doce euros
Pila AA 1.5V	6	11,10	Once euros con diez céntimos
Porta pilas	1	1,50	Un euro con cincuenta céntimos
Motor DC	2	7,90	Siete euros con noventa céntimos
Interruptor	1	0,95	Cero euros con noventa y cinco céntimos
COSTE TOTAL COMPONENTES (€)		**78,34**	**Setenta y ocho euros con treinta y cuatro céntimos**

Tabla 3. Precios unitarios totales

20.2 CUADRO DE PRECIOS UNITARIOS DESCOMPUESTOS

En la siguiente tabla se presentan todos los componentes y dispositivos que se han utilizado en la realización de este proyecto con sus respectivos precios unitarios descompuestos.

COMPONENTES	UNIDADES	PARCIAL (€)	TOTAL (€)
Tablerillo de madera 0.07x0.14m	2	0,95	**1,90**
Tablerillo de madera 0.07x0.11m	1	0,65	**0,65**
Tablerillo de madera 0.07x0.10m	1	0,55	**0,55**
Varilla roscada Diámetro 6mm	4	1,05	**4,20**
Varilla roscada Diámetro 3mm	2	0,65	**1,30**
Rueda Diámetro 6cm	2	5,50	**11,00**
Tuerca hexagonal Diámetro 6mm	16	0,20	**3,20**
Tuerca hexagonal Diámetro 3mm	4	0,12	**0,48**
Tuerca de seguridad Diámetro 6mm	12	0,28	**3,36**
Tuerca de seguridad Diámetro 3mm	4	0,19	**0,76**
Arandela Diámetro 6mm	16	0,15	**2,40**
Arandela Diámetro 3mm	8	0,08	**0,64**
Pletina Diámetro 3mm	2	0,65	**1,30**
Tornillo Diámetro 3mm	4	0,35	**1,40**
Brida	9	0,05	**0,45**
Cable Arduino macho-macho	2	0,08	**0,16**
Cable Arduino macho-hembra	8	0,12	**0,96**
Cable motor driver	4	0,07	**0,28**
IMU MPU-6050	1	3,95	**3,95**
Driver L298N	1	5,95	**5,95**

Arduino UNO	1	12,00	**12,00**
Pila AA 1.5V	6	1,85	**11,10**
Porta pilas	1	1,50	**1,50**
Motor DC	2	3,95	**7,90**
Interruptor	1	0,95	**0,95**
COSTE TOTAL COMPONENTES (€)			**78,34**
I.V.A. (21%) (€)			**16,45**
TOTAL (€)			**94,79**

Tabla 4. Precios unitarios descompuestos

20.3 MANO DE OBRA

En el coste de mano de obra se tiene en cuenta también el tiempo de pruebas, investigación y ensayos realizados para la construcción del prototipo, de esta forma se ha estimado un tiempo total de 250 horas.

Se detalla también el número de personas que han participado en el proyecto, en este caso un único operario, y el sueldo por hora está basado en el que cobra en la actualidad un ingeniero electrónico.

SUELDO INGENIERO ELECTRÓNICO	CANTIDAD (HORAS)	NÚMERO OPERARIOS	COSTE (€)
25€/Hora	250	1	**6250,00**
COSTE TOTAL MANO DE OBRA (€)			**6250,00**
I.V.A. (21%) (€)			**1312,50**
TOTAL (€)			**7562,50**

Tabla 5. Presupuesto de la mano de obra

20.4 LICENCIAS DE SOFTWARE

La licencia de software que es necesaria adquirir para poder realizar el proyecto es la de Matlab y Simulink ya que Arduino es un software libre.

LICENCIA DE MATLAB Y SIMULINK (€)	**250,00**
I.V.A. (21%) (€)	**52,50**
TOTAL (€)	**302,50**

Tabla 6. Presupuesto de las licencias de software

20.5 PRESUPUESTO FINAL

En la siguiente tabla se recogen los tres presupuestos anteriores y el presupuesto final del proyecto.

TOTAL COMPONENTES (€)	**94,79**
TOTAL MANO DE OBRA (€)	**7562,50**
TOTAL LICENCIAS (€)	**302,50**
PRESUPUESTO FINAL (€)	**7959,79**

Tabla 7. Presupuesto final